(2)

RADC-TR-85-37, Vol III (of three)
Final Technical Report
February 1985

AD-A153 990

SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES Software Quality Evaluation Guidebook

Boeing Aerospace Company

Thomas P. Bowen, Gary B. Wigle and Jay T. Tsai

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-85-37, Volume III (of three) has been reviewed and is approved for publication.

APPROVED: Roger B Panara

ROGER B. PANARA
Project Engineer

APPROVED: Raymond P. Urtz Jr

RAYMOND P. URTZ, JR.
Technical Director
Command & Control Division

FOR THE COMMANDER: Donald A Brantingham

DONALD A. BRANTINGHAM
Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COEE) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS N/A
2a SECURITY CLASSIFICATION AUTHORITY N/A 2b DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
4 PERFORMING ORGANIZATION REPORT NUMBER(S) D182-11678-3	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-85-37, Vol III (of three)
6a NAME OF PERFORMING ORGANIZATION Boeing Aerospace Company 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COEE)
6c. ADDRESS (City, State and ZIP Code) P.O. Box 3999 Seattle WA 98124	7b. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b OFFICE SYMBOL (If applicable) COEE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-82-C-0137
8c ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441-5700	10 SOURCE OF FUNDING NOS PROGRAM ELEMENT NO: 63728F PROJECT NO: 2527 TASK NO.: 03 WORK UNIT NO: 05

11 TITLE (Include Security Classification)
SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES Software Quality Evaluation Guidebook

12. PERSONAL AUTHOR(S)
Thomas P. Bowen, Gary B. Wigle, Jay T. Tsai

13a. TYPE OF REPORT	13b. TIME COVERED	14 DATE OF REPORT (Yr., Mo., Day)	15 PAGE COUNT
Final	FROM Aug 82 TO Oct 84	February 1985	314

16 SUPPLEMENTARY NOTATION
N/A

17 COSATI CODES

FIELD	GROUP	SUB. GR
09	02	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Software Quality
Software Quality Metrics

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Volume I (of three) describes the results and presents recommendations for integrating the RADC developed software quality metrics technology into the Air Force software acquisition management process and for changing Air Force acquisition documentation. In addition, changes to the baseline software quality framework are presented and features of a proposed specification methodology are summarized. Terminology and life cycle phases are consistent with the December 1983 draft of the DOD-STD-SDS, Defense System Software Development.

Volume II (of three) describes how the software acquisition manager specifies software quality requirements, consistent with needs. Factor interrelationships, tradeoff among factor quality levels in terms of relative costs and an example for a command and control application are described. Procedures for assessing compliance with the specified requirements based on an analysis of data collected using procedures described in volume III are included.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☒ DTIC USERS ☐	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Roger B. Panara	22b TELEPHONE NUMBER (Include Area Code) (315) 330-4654 22c OFFICE SYMBOL RADC (COEE)

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Volume III (of three) describes procedures and techniques for evaluating achieved quality levels. Worksheets for use in metric data collection by software life cycle phases and scoresheets for scoring each factor are provided.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

PREFACE

This document is the third of three volumes of the Final Technical Report (CDRL A004) for the Specification of Software Quality Attributes contract, F30602-82-C-0137. Contract work was performed by Boeing Aerospace Company (BAC) for Rome Air Development Center (RADC) to provide methods, techniques, and guidance to Air Force software acquisition managers who specify the requirements for software quality.

The purpose of this contract was to (1) consolidate results of previous RADC contracts dealing with software quality measurement, (2) enhance the software quality framework, and (3) develop a methodology to enable a software acquisition manager to determine and specify software quality factor requirements. We developed the methodology and framework elements to focus on an Air Force software acquisition manager specifying quality requirements for embedded software that is part of a command and control application. This methodology and most of the framework elements are generally useful for other applications and different environments.

The Final Technical Report consists of three volumes:

a. Volume I, Specification of Software Quality Attributes—Final Report.
b. Volume II, Specification of Software Quality Attributes—Software Quality Specification Guidebook.
c. Volume III, Specification of Software Quality Attributes—Software Quality Evaluation Guidebook.

Volume I describes the results of research efforts conducted under this contract, including recommendations for integrating quality metrics technology into the Air Force software acquisition management process, recommended changes to Air Force software acquisition documentation, and summaries of software quality framework changes and specification methodology features.

Volumes II and III describe the methodology for using the quality metrics technology and include an overview of the software acquisition process using this technology and the quality framework. Volume II describes methods for specifying software quality requirements and addresses the needs of the software acquisition manager. Volume III

describes methods for evaluating achieved quality levels of software products and addresses the needs of data collection and analysis personnel.

Volume II also describes procedures and techniques for specifying software quality requirements in terms of quality factors and criteria. Factor interrelationships, relative costs to develop high quality levels, and an example for a command and control application are also described. Procedures for assessing compliance with specified requirements are included.

Volume III also describes procedures and techniques for evaluating achieved quality levels of software products. Worksheets for collecting metric data by software life-cycle phase and scoresheets for scoring each factor are provided in the appendixes. Detailed metric questions on worksheets are nearly identical to questions in the Software Evaluation Reports proposed as part of the Software Technology for Adaptable Reliable Systems (STARS) Measurement data item descriptions (DID).

Terminology and life-cycle phases used in the guidebooks are consistent with the December 1983 draft of the Department of Defense software development standard (DOD-STD-SDS) (e.g., the term computer software configuration item (CSCI) is used rather than computer program configuration item (CPCI)).

CONTENTS

FIGURES

TABLES

GLOSSARY

AFCMD	Air Force Contracts Management Division
AFCL	Air Force Logistics Command
AFPRO	Air Force Plant Representative Office
APSE	Ada programming support environment
ASD	Aeronautical Systems Division
CDR	critical design review
CPCI	computer program configuration item
CSC	computer software component
CSCI	computer software configuration item
DACS	Data and Analysis Center for Software
DAE	Defense Acquisition Executive
DID	data item description
DOD	Department of Defense
DOD-STD-SDS	Department of Defense software development standard
DOD-STD-SQS	Department of Defense software quality standard
ESD	Electronic Systems Division
FCA	functional configuration audit
FSD	full-scale development
HOL	high order language
I/O	input/output
IV&V	independent validation and verification
PCA	physical configuration audit
PDL	program design language
PDR	preliminary design review
QA	quality assurance
QM	quality metrics
RADC	Rome Air Development Center
SD	Space Division
SDR	system design review
SPO	System Program Office
SSR	software specification review
STARS	Software Technology for Adaptable Reliable Systems
TRR	test readiness review
V&V	verification and validation

1.0 INTRODUCTION

1.1 BACKGROUND

There has been a recent, increased awareness of critical problems encountered in developing large-scale systems involving software. These problems include cost and schedule overruns, high cost sensitivity to changes in requirements, poor performance of delivered systems, high system-maintenance costs, and lack of reusability.

The government (the Department of Defense (DOD) in particular) as a customer for large-scale system developments, has sponsored efforts to address these problems; for example, development of Ada programming language and Ada programming support environments (APSE), proposed DOD standards for software development (DOD-STD-SDS) and quality (DOD-STD-SQS), the Software Technology for Adaptable Reliable Systems (STARS) program, proposed STARS measurement data item descriptions (DID), and various development aids and tools. These all provide partial solutions.

Since 1976, Rome Air Development Center (RADC) has pursued a program intended to achieve better control of software quality. Through a series of related contracts, this program has sought to identify key software quality issues and to provide a valid methodology for specifying software quality requirements and measuring achieved quality levels of software products released incrementally during the software life cycle. A quality model was established in which a hierarchical relationship exists between a user-oriented quality factor at the top level and software-oriented attributes at the second and third levels (criteria and metrics). Software quality is predicted and measured by the presence, absence, or degree of identifiable software attributes. (See Sec. 2.2 for an explanation of the quality model and an overview of quality factors and attributes.)

The Final Technical Report for this contract (F30602-82-C-0137) contains the most recent results of the RADC software quality program. This report incorporates pertinent results from and uses foundations established in previous contracts. The Final Technical Report consists of three volumes: the Final Report, the Software Quality Specification Guidebook, and the Software Quality Evaluation Guidebook.

1.2 PURPOSE

The purpose of this guidebook (Vol. III, Software Quality Evaluation Guidebook) is to provide a comprehensive set of procedures and techniques to enable data collection personnel to apply quality metrics to software products and to evaluate the achieved quality levels. Volume II, Software Quality Specification Guidebook, provides a comprehensive set of procedures and techniques to enable an Air Force software acquisition manager to specify quality requirements for software embedded in command and control systems. Volume I, Final Report, summarizes the results of contract task efforts.

The purpose of the quality metrics technology is to provide a more disciplined engineering approach to specifying, predicting, and evaluating software quality. The benefits of this approach include software life-cycle cost savings (or cost avoidance) and software products that reflect user-customer quality needs. Rigorous application of metrics at incremental releases of software products throughout the life cycle provides for early detection of quality-related problems. Periodic assessment of quality levels provides better management visibility and enables timely decision making.

1.3 SCOPE

Section 2.0 describes the role of quality metrics in the software acquisition process. Descriptions of the system acquisition life cycle and software development cycle are provided with a discussion of their relationships. Specifying quality requirements and monitoring software product quality levels are described within the life-cycle perspective. The software quality model and framework elements are introduced.

Section 3.0 describes quality framework terminology and concepts key to understanding subsequent details. All framework elements—factors, criteria, metrics, worksheets, and scoresheets—are also described.

Section 4.0 describes procedural steps for applying quality metrics to products of the development process, calculating achieved quality levels, and analyzing scoring results.

This guidebook incorporates pertinent results from previous research concerning software quality measurement conducted for RADC. Results of this research are described in Software Quality Measurement for Distributed Systems, RADC-TR-83-175, Volumes I, II, and III. Significant enhancements were made to framework elements during this contract. Factors are categorized under performance, design, and adaptation to aptly indicate acquisition concerns. Criteria are organized under the same three acquisition concerns, thereby simplifying the attribute relationships. Metric questions on the worksheets include explanatory information and formulas and are nearly identical to the questions in the Software Evaluation Reports proposed as part of the STARS measurement DIDs. Software life-cycle phases and terminology used throughout this guidebook are consistent with the December 1983 draft of DOD-STD-SDS.

1.4 USE OF THE GUIDEBOOKS

This Software Quality Evaluation Guidebook addresses the needs of personnel collecting and analyzing metric data. Procedures are provided for applying metrics, generating metric scores, analyzing scoring, and reporting results. The Software Quality Specification Guidebook (see Vol. II) addresses the needs of Air Force software acquisition managers. Procedures are provided for specifying quality requirements and for assessing compliance with requirements.

Procedures in each guidebook are contained in Section 4.0. Sections 1.0, 2.0, and 3.0 contain nearly identical information on the elements, perspective, and role of quality metrics technology.

The guidebooks were designed for use with new projects, in which procedures are performed (primarily) chronologically throughout system and software life cycles as depicted in Figure 1.4-1. Using quality metrics technology and guidebooks for evaluating system and software products in other contexts is addressed in Section 4.0. Detailed explanations of life-cycle phases, review points, framework elements, and methodology are provided in Sections 2.0, 3.0, and 4.0.

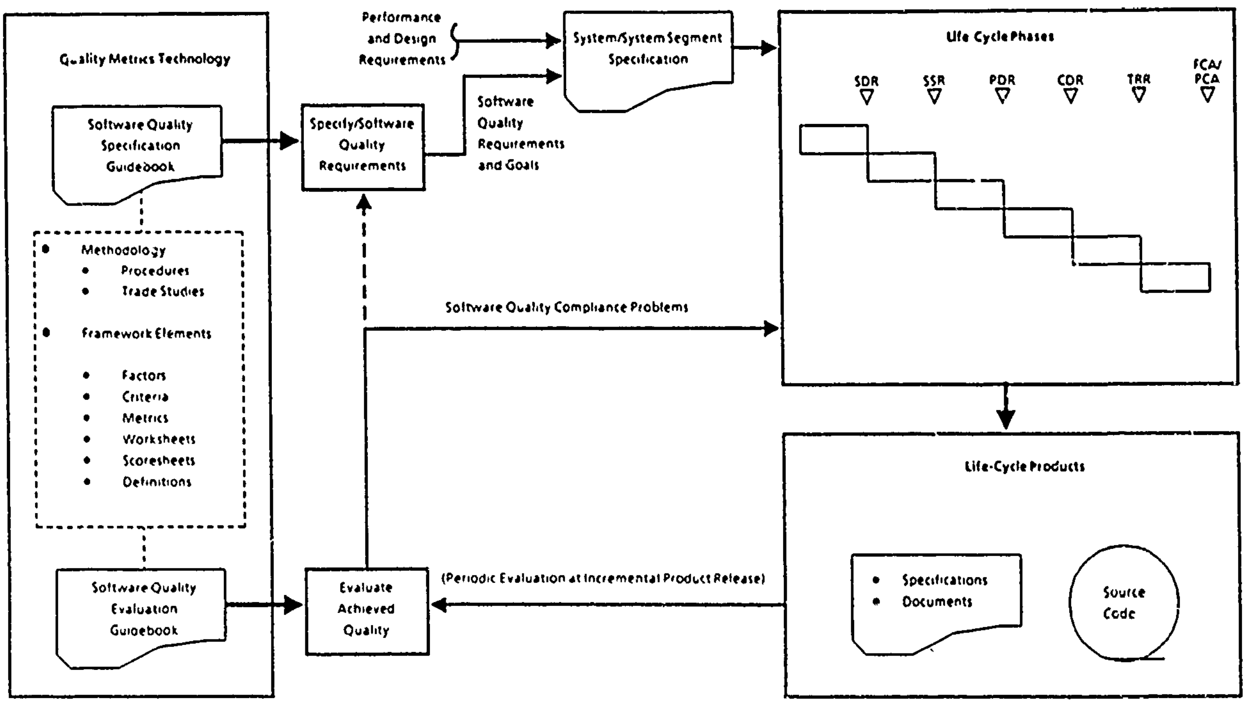


Figure 1.4-1 Software Quality Measurement Methodology

2.0 ROLE OF QUALITY METRICS IN THE SOFTWARE ACQUISITION PROCESS

This section examines elements of Air Force system acquisition and software acquisition processes, describes the process used for specifying and monitoring quality levels, and discusses the role of quality metrics (QM) technology in the Air Force software acquisition management process. Considerations include how QM technology can be integrated into the Air Force software acquisition process and how existing mechanisms within the acquisition process can be used to implement QM technology. Advantages and disadvantages of using QM technology in software acquisition management and of integrating QM technology into the software acquisition management process are also discussed.

2.1 SOFTWARE ACQUISITION PROCESS

The following sections describe selected concepts associated with Air Force software acquisition management, including system acquisition life cycle, software development cycle, life-cycle relationships, software acquisition management, verification and validation (V&V), and quality assurance (QA). Concepts introduced here provide a basis for discussions of QM technology integration and implementation in the acquisition process in later sections. The system acquisition life cycle and software development cycle are fully defined in DODD 5000.1 and DOD-STD-SDS and are only summarized here. This section is not intended to describe all activities of each life-cycle phase but to establish the background for discussion of the role of QM technology.

2.1.1 System Acquisition Life Cycle

The system acquisition life cycle defined in DOD-STD-SDS consists of four phases: concept exploration, demonstration and validation, full-scale development (FSD), and production and deployment. Four major decision points are associated with these phases as shown in Figure 2.1-1 and as defined in DODD 5000.1 (Major System Acquisition). These points are mission need determination; concept selection, milestone I; program go-ahead, milestone II; and production and deployment, milestone

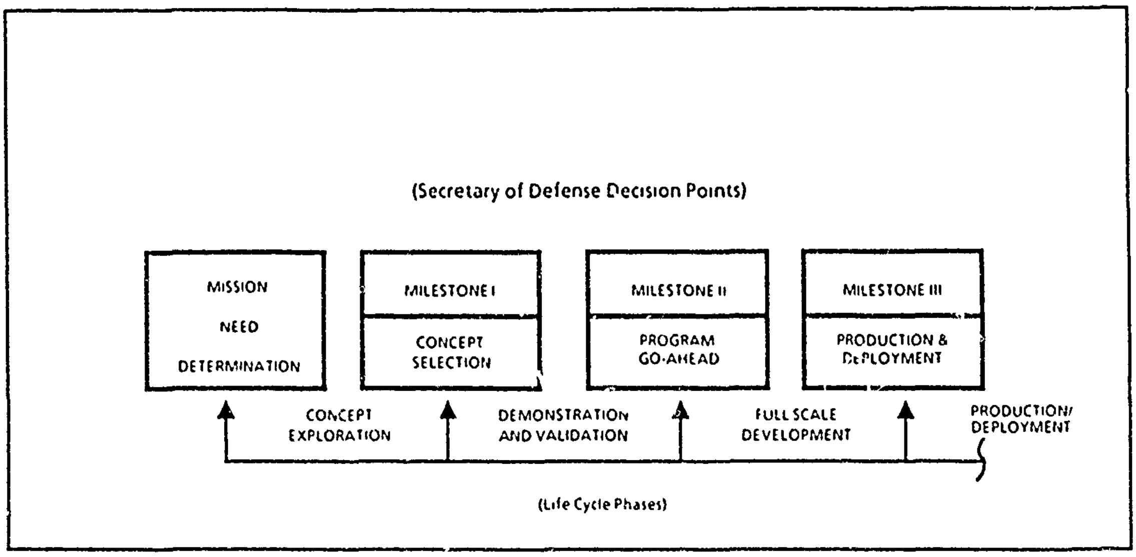


Figure 2.1-1 System Acquisition Life-Cycle Phases and Decision Points

III. The Secretary of Defense, advised by the Defense Acquisition Executive (DAE), decides at these points whether to continue the program and proceed to the next phase or to terminate the program. The system acquisition life cycle applies to the whole system, not the individual parts.

Concept exploration is the initial planning phase, during which the role of and plans for using computer resources in the system are explored. During demonstration and validation, translating operational requirements into functional, interface, and performance requirements is completed; and requirements for each hardware and software configuration item are defined. During FSD, the system is designed, built, tested, and evaluated. These initial three phases should result in a system meeting specified requirements. Production and deployment includes production (if applicable) and delivery and includes all activities involved in supporting the system until it is retired.

2.1.2 Software Development Cycle

The software development cycle, as defined in DOD-STD-SDS, consists of six phases: software requirements analysis, preliminary design, detailed design, coding and unit testing, computer software component (CSC) integration and testing, and computer software configuration item (CSCI) level testing (see Fig. 2.1-2). This cycle, however, is not standardized and there are many variations throughout the industry. Although names and breakdowns vary, the same process is generally followed.

All software requirements are specified during software requirements analysis. The authenticated software requirements specification (signed off by both the customer and contractor) forms the baseline for preliminary design. During preliminary design, a modular, top-level design is developed from the software requirements. During detailed design, the top-level design is refined to successively lower levels until individual units, which perform single, nondivisible functions, are defined. During coding and unit testing, the designer translates the design approach into code and executes verification tests. During CSC integration and testing, code units are integrated and informal tests are performed on aggregates of integrated units. This cycle concludes with CSCI-level testing, during which formal tests are conducted on the software.

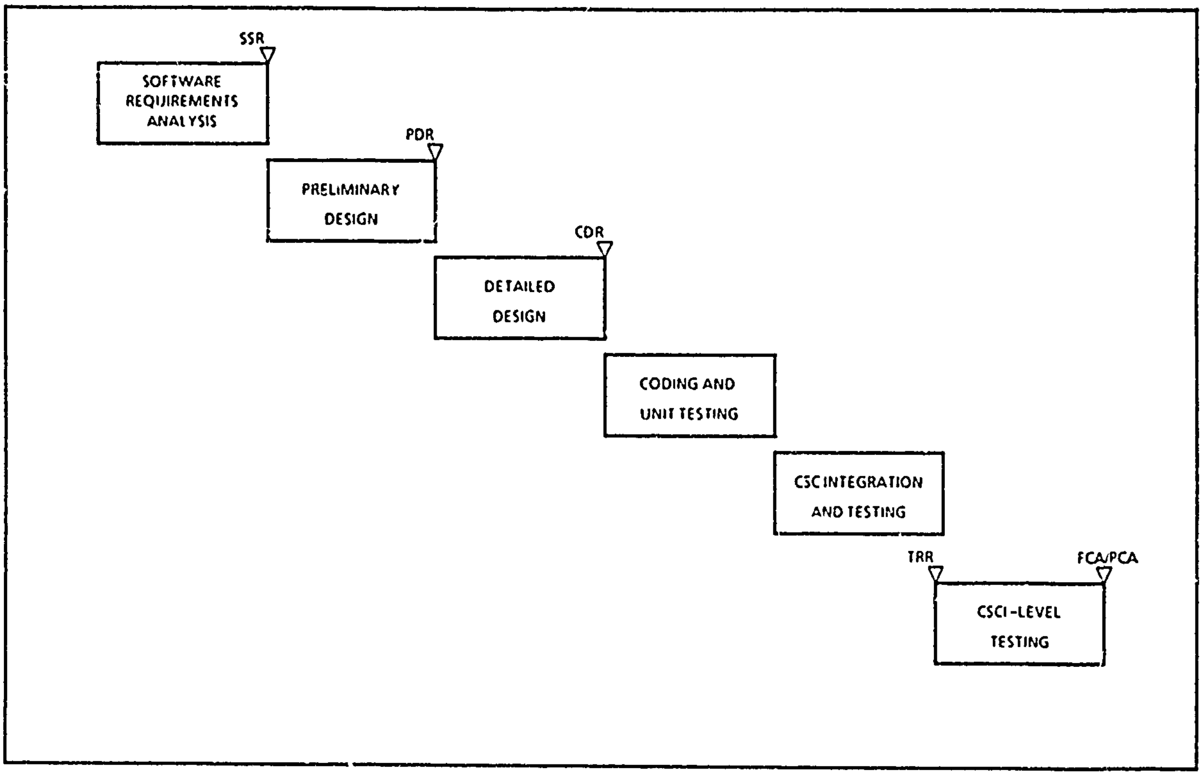


Figure 2.1-2 Software Development Cycle

As with the system acquisition life cycle, the software development cycle has decision points associated with most phases. These decision points (shown in Fig. 2.1-2) are the: software specification review (SSR), preliminary design review (PDR), critical design review (CDR), test readiness review (TRR), and functional configuration audit (FCA)/physical configuration audit (PCA). These decision points are quite different from decision points associated with the system acquisition life cycle. At these decision points it is not determined whether to continue or terminate the program; rather, progress up to that point is reviewed and it is decided if the developer has completed the current phase and is ready to proceed into the next phase.

2.1.3 Life-Cycle Relationships

Each CSCI to be developed goes through the entire software development cycle. The software development cycle can be completed in a single phase of the system acquisition life cycle or can overlap several phases. For example, software could be developed for risk-reduction analysis during concept exploration or demonstration and validation. This software could be used to validate the feasibility of an algorithm or to compare alternative approaches. This type of software may not be in the language required for the operational software and may not be targeted for the same computer. However, it still goes through the entire development cycle. The same is true for test software developed to aid in validation of the operational software. Operational software development may overlap several system life-cycle phases; requirements definition for operational software begins early in the system acquisition life cycle, although operational software is not fully developed until FSD. In this guidebook operational software quality is the primary concern; therefore, the relationship of the operational software development cycle to the system acquisition life cycle will be examined.

There is a specific relationship between the operational software development cycle and the system acquisition life cycle in most system procurements (see Fig. 2.1-3). The software requirements analysis phase overlaps part of the demonstration and validation phase and the beginning of FSD. The remaining operational software development phases occur during FSD; i.e., preliminary design through CSCI-level testing of the software development cycle. This relationship is assumed for the remaining discussions.

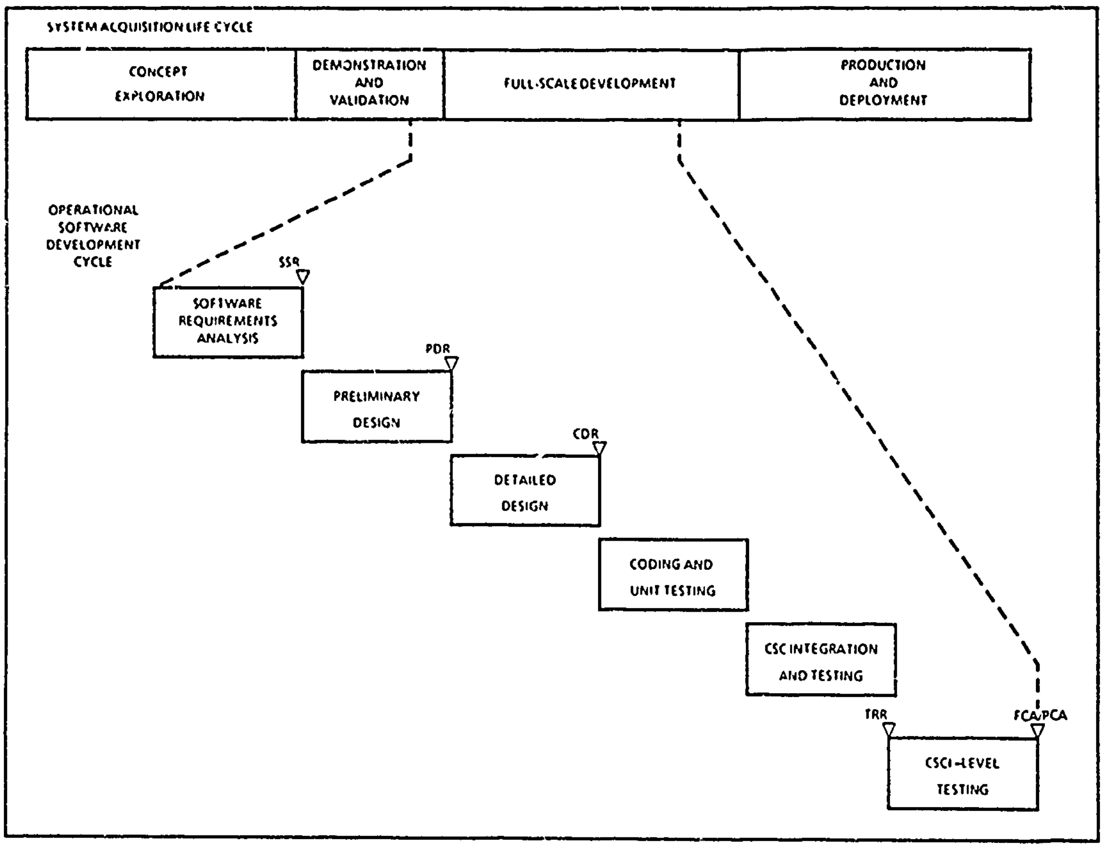


Figure 2.1-3 Life-Cycle Relationship between the System and the Operational Software

2.1.4 Software Acquisition Management

The software acquisition manager has various responsibilities during the software development cycle. This section focuses on two general functions of software acquisition management: (1) specifying requirements and (2) monitoring development to ensure satisfiying the requirements. To describe all that this manager does during the software life cycle is beyond the scope of this guidebook.

Specification of software requirements begins with development of the system specification and continues until all requirements for each CSCI have been specified during software requirements analysis in the software development cycle. These requirements include more than traditional functional and performance requirements. They also include interface, human engineering, language, data base, delivery, self-test, anomaly management, resource reserves, and quality requirements. Many decisions are made to specify these requirements.

The software acquisition manager becomes involved at the system level, when system functional tasks are allocated to software or to hardware. Allocation decisions may be based on trade studies, system engineering, and risk analyses. Once the allocation of functional tasks is completed, specific software requirements can be identified. The result is a set of software capabilities, performance levels, and design constraints. Identification of these specific requirements usually involves decisions supported by trade studies. Such trade studies may include, for example, higher order language (HOL) versus assembly language, distributed processing versus centralized processing, growth capability required for timing and sizing, the degree of human operator interaction required, and efficiency versus maintainability. These software trade studies consider life-cycle costs, risk, schedule, capabilities, software performance, and final product quality. These activities are concluded when the System Program Office (SPO) authenticates (signs off) the software requirements specifications for each CSCI.

Once software requirements are specified, the acquistion manager begins monitoring software development. Monitoring continues throughout preliminary design, detailed design, coding and unit testing, CSC integration and testing, and CSCI-level testing and may continue into the system integration and testing that follows. The primary

concern of monitoring, other than schedule or cost, is whether the software satisfies the requirements. Monitoring provides the acquisition manager with visibility of the evolving product in order to track technical progress and quality. This visibility is achieved through various reviews, audits, documentation, and products required periodically throughout development. Established criteria and measurement methods for each review and audit and for all documentation and products are nescessary for tracking progress. Tracking enables the manager to identify problems early enough to correct them. Two activities providing feedback are V&V and QA.

2.1.5 Verification and Validation

The purpose of V&V is to provide the Air Force with systematic assurance that acquired software will perform missions in accordance with requirements. The terms verification and validation are often used interchangeably, but in the software development cycle distinct concepts are associated with each. The meaning of these terms as used here is as follows:

Verification is the iterative process of determining whether the product of each software development phase fulfills requirements levied by the previous phase. That is, (1) software requirements are verified to ensure that they fulfill system-level requirements, (2) the software design is verified to ensure that it satisfies requirements in the software requirements specification, and (3) code is verified to ensure that it complies with the top-level design and detailed design documents. This process does not consider whether system-level software requirements are correct or whether they actually satisfy users needs.

Validation is a continuing process to ensure that requirements at various levels are correct, thus satisfying mission requirements defined by the using command. Sometimes validation is considered to be the system-level test activity that validates the CSCI against software and system requirements. In reality, it is much more than that. Validation, like verification, continues throughout the software life cycle. For example, when software requirements are allocated and derived, a system-level requirement could be found to be vague or incorrect; or during design, it could be discovered that a software requirement is infeasible or ambiguous. Feedback to the

manager enables corrective action to be taken early in development, thereby reducing risk and cost.

The concept of V&V and its relationship to software development products is shown in Figure 2.1-4. V&V provides feedback to the software acquisition manager concerning software technical performance. The term IV&V is used when V&V is done for the Air Force by a contractor other than either the prime contractor or the subcontractor who is developing the software.

2.1.6 Quality Assurance

According to MIL-S-52779A, the purpose of software QA is to ensure that the software delivered under a contract complies with contract requirements. This type of QA program will not ensure development of a high-quality software product unless software quality attributes are specified in measurable terms as part of the contract. The objective of current QA programs is to provide feedback to the acquisition manager concerning various aspects of the development process. QA is similar to V&V, the major difference being that V&V provides technical feedback on software products at only a few points in time, whereas QA provides feedback on a wide range of development activities. But contractual software quality is not normally defined in quantitative terms. The current goal is simply to achieve better quality through controlling the development processes.

Section 2.3 explores how QM technology can help to expand the scope of QA programs to include specification of software quality requirements and measurement of achieved quality levels for software development products. The following paragraphs explain the current scope of QA programs.

At one time, software QA was equated to testing. As an illustration, Section 4 of the CPCI development specification (according to MIL-STD-483) was called Quality Assurance Provisions. However, as with other products, it was learned that quality cannot be tested into software. Because of cost and schedule impacts, it is usually too late to make changes when quality problems are found during testing. Quality can be affected by how code is written and how software is designed. If a software quality problem is found during testing, it is usually very expensive to redesign and to change

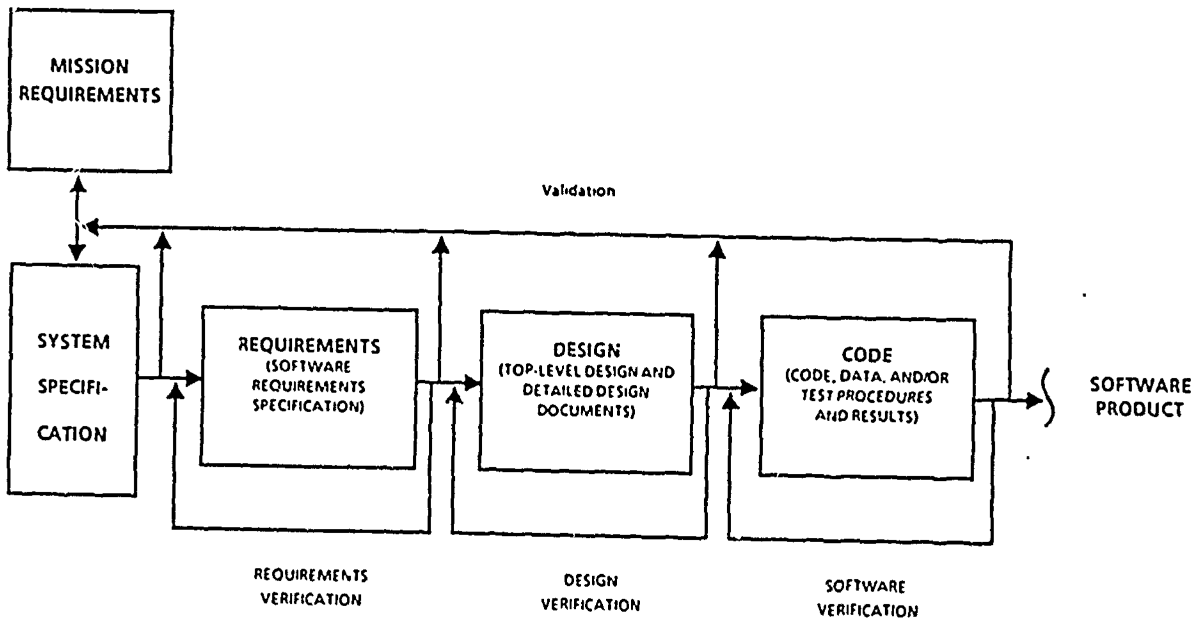


Figure 2.1-4 Relationship of Software Development and V&V

the code. Quality should be planned, designed, and built into software. This realization has lead to the current life-cycle-oriented QA approach. This approach focuses attention on all phases of the software development cycle; and software QA now includes many activities, such as ensuring that software is being developed in accordance with plans, that requirements are traceable, that design and code are easily and economically supportable, and that testing is accomplished as planned. These activities provide necessary feedback to the software acquisition manager.

Software quality assurance programs, however, are primarily administrative rather than technical. For example, the QA organization does not trace requirements but ensures that Engineering has developed traceability matrices. The QA function is essentially a checkoff function applied during the software development process; i.e., QA ensures that everything is done as planned. Software QA continues throughout the software development cycle (see Fig. 2.1-5).

Software QA is an evolving discipline. Experience has provided insight into which development practices tend to produce a higher quality software product, and the QA program ensures that selected practices are used by checking the development process. The next step to improving quality is to quantitatively specify quality requirements and to measure and control the quality of the software product as it evolves. Implementing QM technology in the Air Force acquisition process will provide the added dimension of quantitative measures to addressing quality concerns for software products.

2.2 QUALITY METRICS

The purpose of QM technology is to enable the software acquisition manager to specify a desired software quality level for each quality factor of importance to the application and to quantitatively measure the achieved levels of quality at specific points during development. These periodic measurements enable an assessment of current status and a prediction of quality level for the final product. Some problems with delivered software products have been that these products are (to varying degrees) unreliable, incorrect, and/or unmaintainable. QM technology addresses these and other quality-oriented problems by providing a means to specify quality requirements, to quantitatively measure quality achieved during development, and to predict a quality level for the final product.

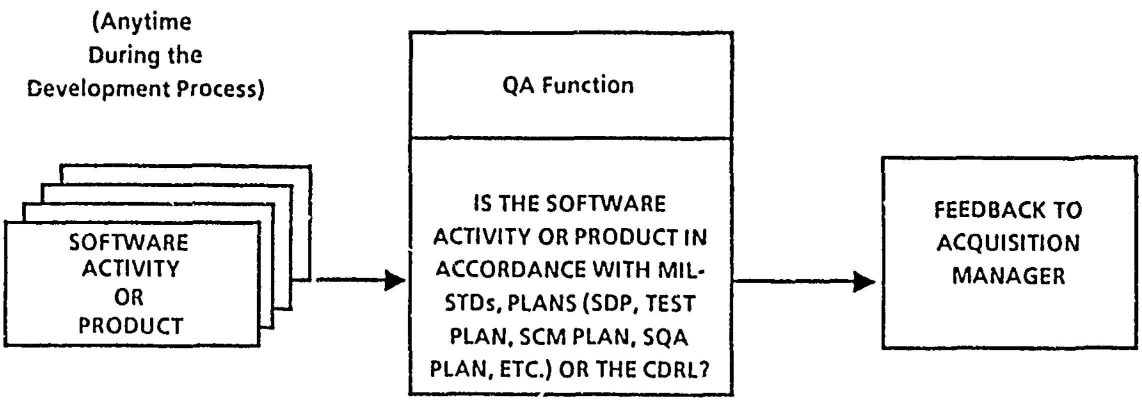


Figure 2.1-5 Software QA Function

QM technology measures the degree of software quality, not the level of software technical performance; e.g., how easy is it to maintain the software, not how accurate is the navigation algorithm. However, the process of specifying and measuring quality levels is analogous to the process of specifying and measuring technical performance. Both processes begin with similar activities: system needs are assessed, trades are performed (involving resources and levels of performance or levels of quality), and requirements are specified. Subsequent phases involve evaluations of how well these requirements are being satisfied.

Technical performance levels are traditionally evaluated by modeling in early development stages and by testing in later development stages. Quality has traditionally been evaluated by such methods as reviews, walkthroughs, and audits. This type of quality evaluation ensures that, for example, designs are traceable to requirements, configuration management is adequate, and standards and plans are being followed. However, it does not address such quality issues as software reliability, correctness, and maintainability. QM technology enables a quantitative assessment of these types of quality factors at different stages of development, thereby ensuring that specified quality levels are being satisfied in a manner similar to performance evaluation by testing.

Figure 2.2-1 depicts the software life-cycle model used in QM technology. The software model is shown in typical relationship to two system acquisition phases. Eight development states are shown with typical review and audit points. There are two system-level activities involving software: system/software requirements analysis and system integration and testing (both shown in dashed boxes). (Operational testing and evaluation is the last FSD phase but is not shown as it is normally not performed by the development contractor.) There are six software development phases: software requirements analysis, preliminary design, detailed design, coding and unit testing, CSC integration and testing, and CSCI-level testing. These phases refer to the same development activities as are described in Section 2.1. This division of activities was chosen because at the end of each activity shown in Figure 2.2-1 a configuration baseline generally is established, and software products (specifications, documents, code) describing that baseline are available for review or audit and the application of metric measurements. Also illustrated in Figure 2.2-1 are the two points at which quality requirements are specified and the eight points at which quality levels are measured (monitored). These measurement points generally correspond to the review or audit points for configuration baselines.

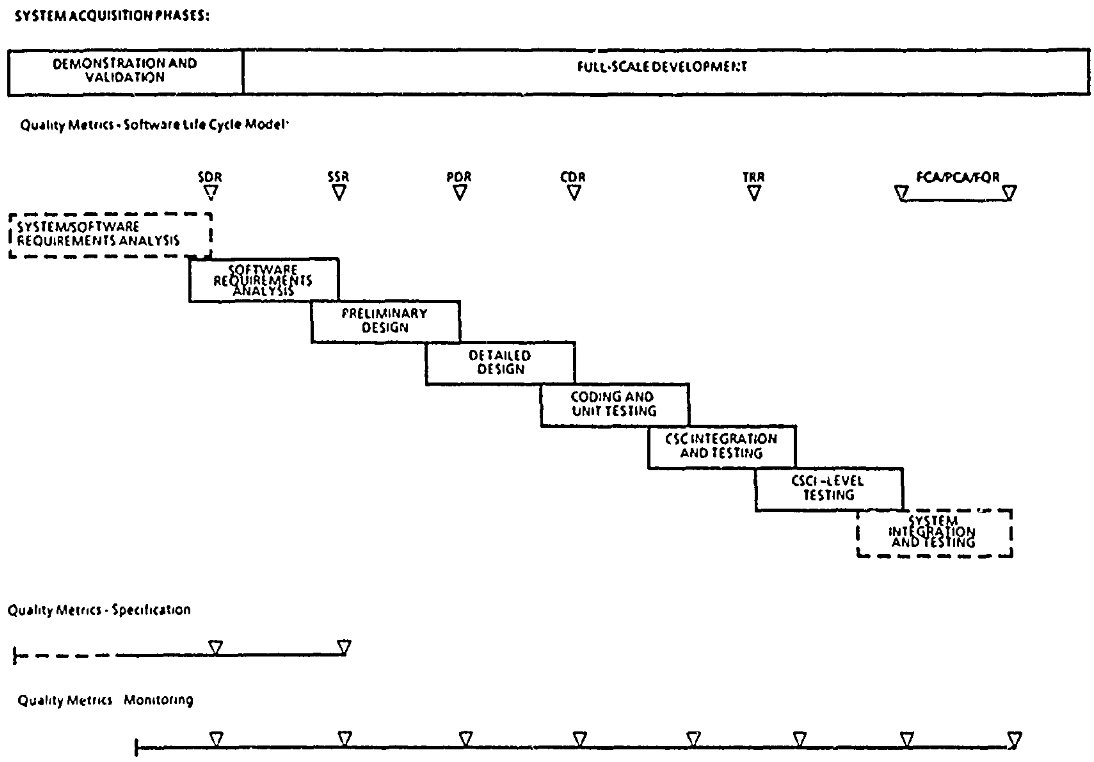


Figure 2.2-1 Quality Metrics Technology - Life-Cycle Model

2.2.1 Framework

A hierarchical model for quality has been established (see Fig. 2.2-2). User-oriented factors (e.g., reliability, correctness, maintainability) are at the top level, software-oriented criteria are at the next level, and metrics—quantitative measures of characteristics—are at the lowest level.

This model is flexible in that it indicates a general relationship between each factor and its attributes. This permits updating of individual elements to reflect technology advances without affecting the model itself. For example, as new user concerns evolve, new factors can be added at the top level; and as software technology evolves, criteria and metrics can be added, deleted, or modified as necessary. There are currently 13 quality factors, 29 criteria, 73 metrics, and more than 300 metric elements (distinct parts of a metric). Table 2.2-1 shows the 13 quality factors and describes the primary user concern for choosing each factor. Quality factors and user concerns are categorized by three types of acquisition concerns with respect to the software: (1) product performance—how well does the software function in its normal environment; (2) product design—how valid (appropriate) is the design with respect to requirements, verification, and maintenance; and (3) product adaptation—how easy is it to adapt the software for use beyond its original intended use (e.g., for new requirements, a new application, or a different environment).

Figures 2.2-3, 2.2-4, and 2.2-5 show the quality factors, criteria, and metrics in the hierarchical relationships of the software quality model. The metrics are identified by acronym only in the figures. These and other framework elements for QM technology are described in detail in Section 3.0. The following sections describe some aspects involved in specifying and monitoring software quality using QM technology.

2.2.2 Quality Specification

When determining and specifying software quality requirements, system needs are assessed from a quality perspective; the desired quality factors, associated criteria, and applicable metrics are selected; and quality-level goals are derived for each separate quality factor. When assessing system needs, application characteristics should be considered. For example, if the system will have a long life cycle, emphases

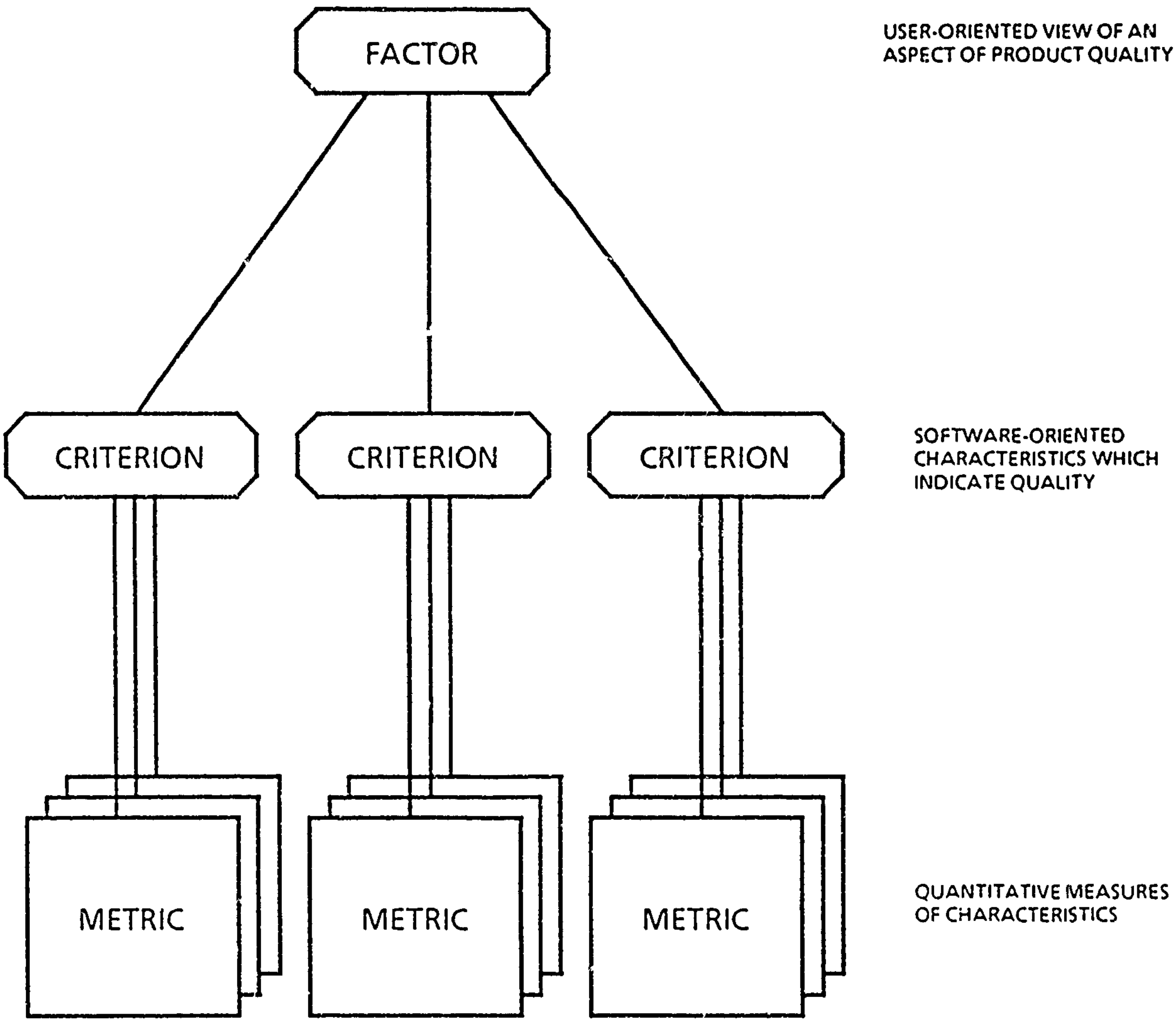


Figure 2.2-2 Software Quality Model

Table 2.2-1 Quality Concerns

Acquisition Concern	User Concern	Quality Factor
PERFORMANCE - HOW WELL DOES IT FUNCTION?	HOW WELL DOES IT UTILIZE A RESOURCE?	EFFICIENCY
	HOW SECURE IS IT?	INTEGRITY
	WHAT CONFIDENCE CAN BE PLACED IN WHAT IT DOES?	RELIABILITY
	HOW WELL WILL IT PERFORM UNDER ADVERSE CONDITIONS?	SURVIVABILITY
	HOW EASY IS IT TO USE?	USABILITY
DESIGN - HOW VALID IS THE DESIGN?	HOW WELL DOES IT CONFORM TO THE REQUIREMENTS?	CORRECTNESS
	HOW EASY IS IT TO REPAIR?	MAINTAINABILITY
	HOW EASY IS IT TO VERIFY ITS PERFORMANCE?	VERIFIABILITY
ADAPTATION - HOW ADAPTABLE IS IT?	HOW EASY IS IT TO EXPAND OR UPGRADE ITS CAPABILITY OR PERFORMANCE?	EXPANDABILITY
	HOW EASY IS IT TO CHANGE?	FLEXIBILITY
	HOW EASY IS IT TO INTERFACE WITH ANOTHER SYSTEM?	INTEROPERABILITY
	HOW EASY IS IT TO TRANSPORT?	PORTABILITY
	HOW EASY IS IT TO CONVERT FOR USE IN ANOTHER APPLICATION?	REUSABILITY

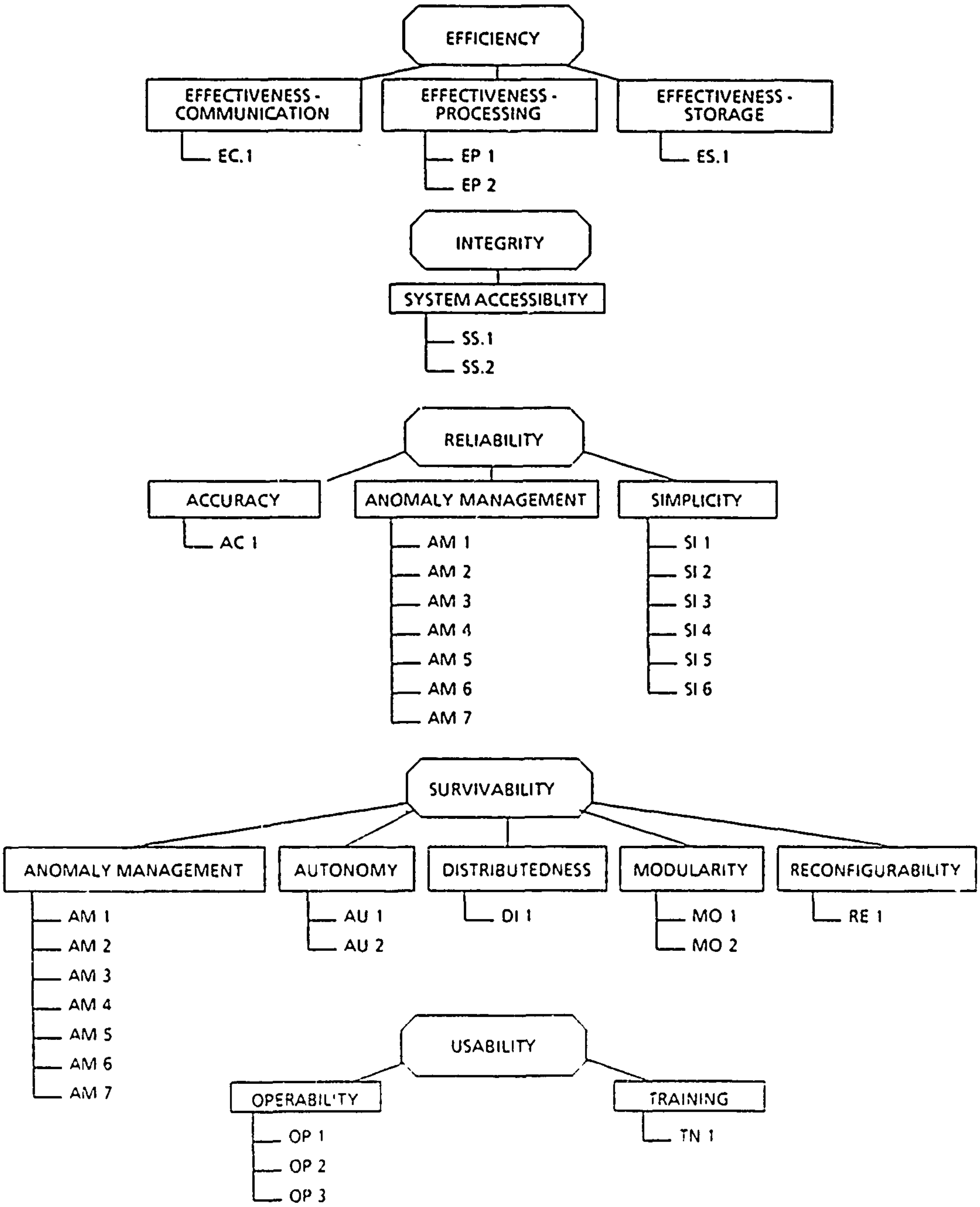


Figure 2.2-3 Performance Factor Attributes

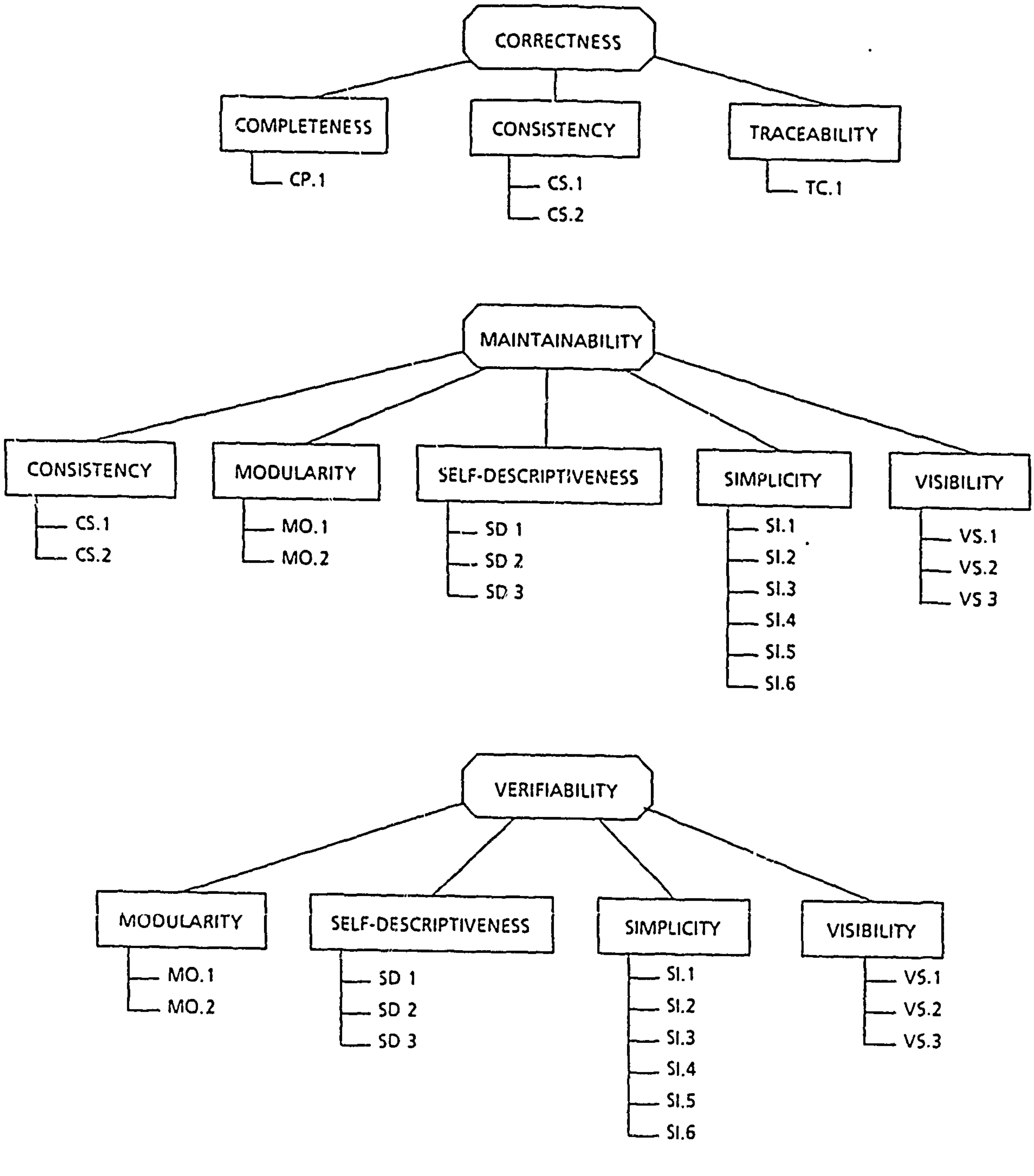


Figure 2.2-4 Design Factor Attributes

35

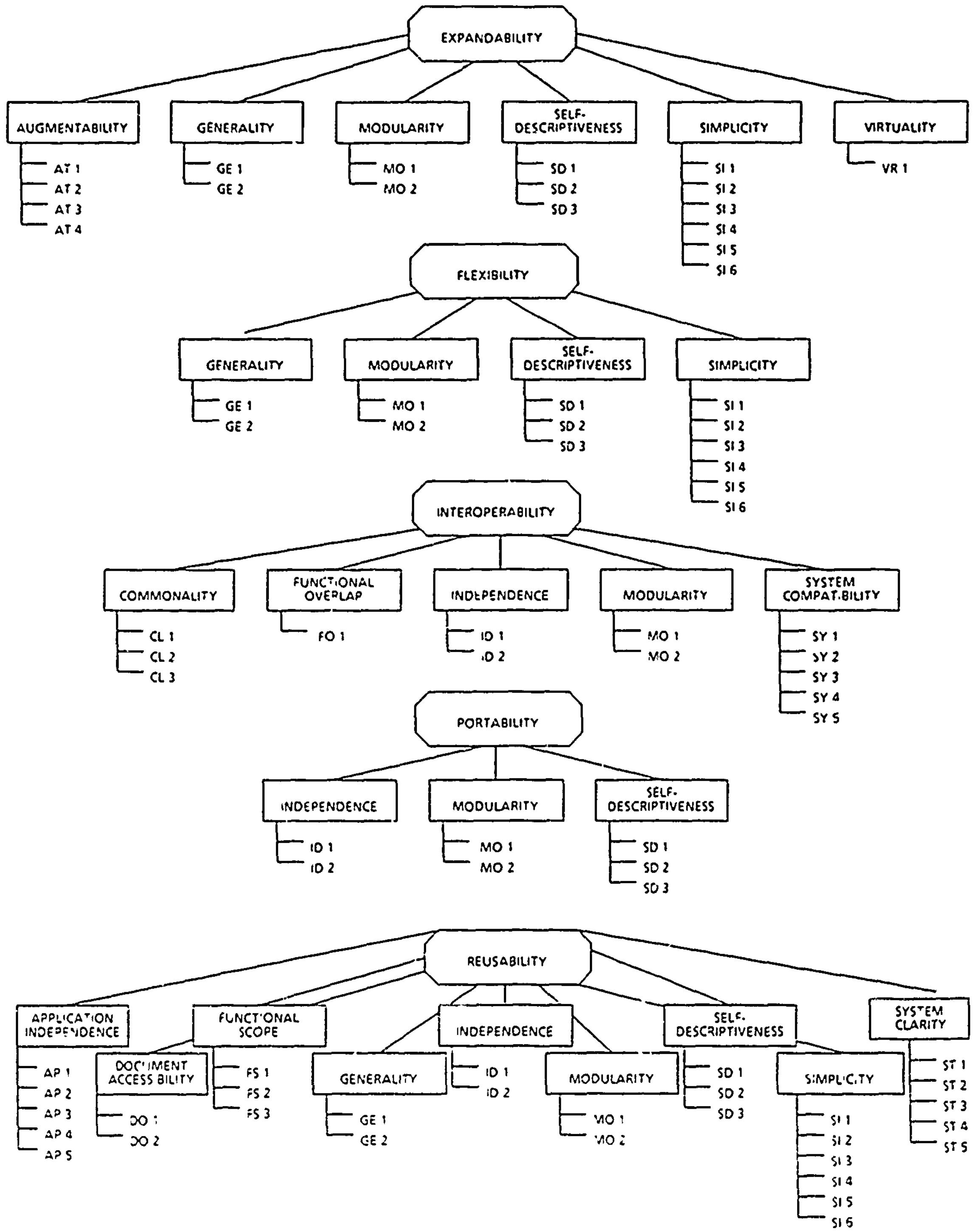


Figure 2.2-5 Adaptation Factor Attributes

on maintainability, flexibility, portability, and expandability are recommended. Factor goals define the required quality levels to be achieved for the factor (i.e., excellent, good, or average). In general, choosing a higher quality goal will result in more resources being expended to achieve that level. When deriving factor goals, interrelationships between factors should be considered because a high quality goal for one factor may conflict with a high quality goal for another factor. Table 2.2-2 shows the beneficial and adverse relationships between quality factors; some factors have a positive relationship and others conflict. For example, specifying a high quality level for most factors conflicts with specifying a high quality level for efficiency.

A typical problem for an embedded software system arises when reliability is of the utmost importance because of the type of mission to be performed, but efficiency is also required because of space and weight limitations, and flexibility is needed because of the variety of missions and/or targets. It is normally infeasible to select and achieve high quality levels for all three factors. Highly efficient code is usually tightly written assembly-level code and tends to be not as reliable or as amenable to changes (flexible) as looser, more structured HOL code. And code written to be reliable and flexible tends to be less efficient. Trade studies are needed to resolve these problems. If some efficiency is sacrificed for reliability, then performance goals (e.g., for accuracy or range) may be affected. If some flexibility is sacrificed for efficiency, then the scope of the missions and/or targets may be reduced. QM technology provides an aid for decision making when selecting quality-level goals, when determining feasible software requirements, and for allocating acquisition resources. Several iterations of quality tradeoffs may be required for choosing reasonable quality goals. Section 4.0 of the specification guidebook (Vol. II) provides specific techniques for choosing quality factors and includes consideration of application characteristics and factor interrelationships.

2.2.3 Quality Monitoring

When monitoring software quality, the quality metrics (in the form of questions on worksheets) are applied to software products (specifications, documents, code) at different stages of the development cycle, and a quality-level score is calculated for each factor. The factor score predicts a quality level for the final product. The points in the development cycle where data gathering and analysis are recommended is

Table 2.2-2 Software Quality Factor Interrelationships

ACQUISITION CONCERN	QUALITY FACTOR SPECIFIED \ QUALITY FACTOR AFFECTED	PERFORMANCE: EFFICIENCY	PERFORMANCE: INTEGRITY	PERFORMANCE: RELIABILITY	PERFORMANCE: SURVIVABILITY	PERFORMANCE: USABILITY	DESIGN: CORRECTNESS	DESIGN: MAINTAINABILITY	DESIGN: VERIFIABILITY	ADAPTATION: EXPANDABILITY	ADAPTATION: FLEXIBILITY	ADAPTATION: INTEROPERABILITY	ADAPTATION: PORTABILITY	ADAPTATION: REUSABILITY
PERFORMANCE	EFFICIENCY							▽	▽				▽	
	INTEGRITY	▽												
	RELIABILITY	▽				△								
	SURVIVABILITY	▽	▽			△		△		▽	▽		▽	▽
	USABILITY	▽						△	△					
DESIGN	CORRECTNESS							△	△	△	△			△
	MAINTAINABILITY	▽							△	△	△			△
	VERIFIABILITY	▽												
ADAPTATION	EXPANDABILITY	▽	▽	▽	▽							△		
	FLEXIBILITY	▽	▽	▽	▽							△		
	INTEROPERABILITY	▽	▽											
	PORTABILITY	▽												
	REUSABILITY	▽	▽	▽	▽			△				△	△	

△ = POSITIVE EFFECT

▽ = NEGATIVE EFFECT

BLANK = NONE OR APPLICATION DEPENDENT

shown in Figure 2.2-1. These points generally correspond to normal reviews and audits conducted when a configuration baseline has been established (SDR, SSR, PDR, CDR, TRR, and FCA/PCA). Before each review or audit, the metrics selected for the project are applied to software products resulting from that phase of development. This results in a quantitative value for each metric. The metric values are then used to calculate scores for each criterion, and the criteria scores are used to calculate a score (predicted quality level) for each factor.

The quality metrics are applied at incremental points during the development phases. This enables periodic review of progress in meeting quality goal requirements and aids in pinpointing areas of weakness (and strength) in product quality as the product evolves. There are two types of metrics—anomaly detecting and predictive. Both are used in scoring. A low score for predictive metrics indicates that a low score will probably result for the end product because the design is not considering aspects important to achieving the desired quality level. For example, if the design has very little spare storage capacity, the end product will not be highly expandable. A low score for anomaly-detecting metrics indicates an actual design or code deficiency. For example, if provisions are not made for immediate indication of an access violation, software integrity would be jeopardized. Evaluating low metric scores provides an opportunity for identifying deficiencies and anomalies during development when they are more easily corrected.

Worksheets have been devised to help gather metric data. There is a separate worksheet for each development phase, and each worksheet lists only metrics applicable to that phase. A more detailed explanation of the worksheets is provided in Section 3.4. Data collection and analysis are addressed in Section 4.0.

2.3 SOFTWARE ACQUISITION USING QUALITY METRICS

Two general functions of the software acquisition manager are described in Section 2.1.4: (1) specifying requirements and (2) monitoring development to ensure that requirements are being satisfied. Also two general functions associated with QM technology are described in Sections 2.2.2 and 2.2.3: (1) specifying quality requirements and (2) monitoring development to ensure that metric scores are predicting specified quality goals. When using QM technology, monitoring begins

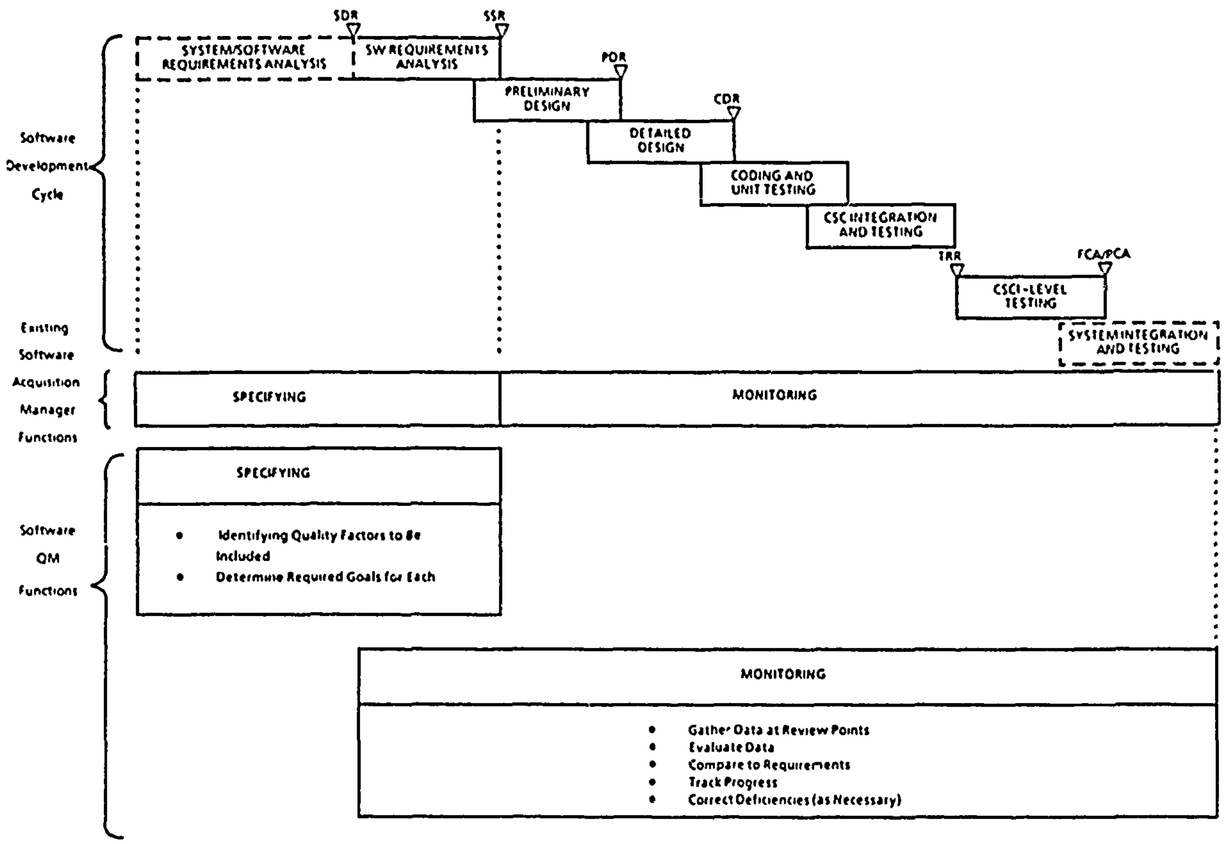


Figure 2.3-1 Software Acquisition Quality Metrics Functions

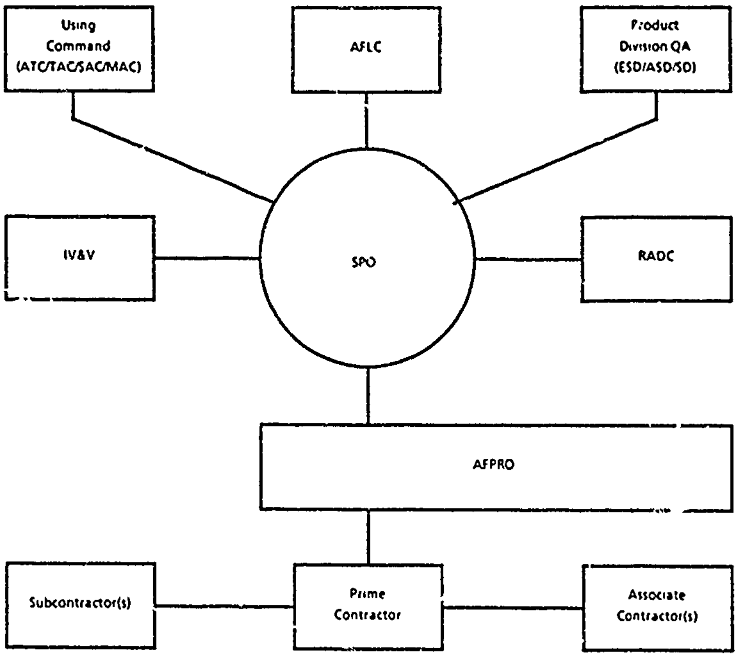
earlier in the development cycle. The relationship of these functions to the software life cycle is shown in Figure 2.3-1.

Specifying and monitoring have not usually overlapped. The specification of software requirements was normally completed before development monitoring began, as shown in Figure 2.3-1. Metric questions have been devised to enable evaluation of software quality reflected in the system specification available at the system design review (SDR). This moves the start of monitoring forward so that the two functions overlap.

Several organizations normally are involved in performing these two functions. Although the internal structure of the Air Force Product Divisions (ESD, ASD, and SD) may differ, the relationship of the SPO to external organizations is basically the same for each division. Organizations that may be involved in the QM functions and their recommended relationships are shown in Figure 2.3-2. Organizational relationships are discussed in the following paragraphs.

Several organizations should be involved in the specification function. The primary organization responsible for software requirements specification is SPO Software Engineering. However, SPO software engineers need help from both the using command and Air Force Logistics Command (AFLC) to fully define software quality needs. Both organizations have a vested interest in requirements affecting system operation and support.

The using command is primarily interested in operational requirements and is especially qualified to contribute to a definition of quality needs for the performance quality factors (e.g., efficiency, integrity, and reliability). AFLC is primarily interested in support requirements and is especially qualified to contribute to a definition of quality needs for the design and adaptation quality factors (e.g., maintainability, expandability, and portability). With input from these organizations, SPO Software Engineering can determine the contractual statement of quality requirements. In addition, the Product Division Software QA organization is normally tasked to ensure that quality requirements are included in the contract. These responsibilities and relationships for the specification function are shown in Figure 2.3-3.



Legend

ATC	= Air Training Command	AF	= Air Force	ESD	= Electronics System Division
TAC	= Tactical Air Command	AFLC	= AF Logistics Command	ASD	= Aeronautical System Division
SAC	= Strategic Air Command	AFPRO	= AF Plant Representative Office	SD	= Space Division
MAC	= Military Air Command	SPO	= System Program Office	QA	= Quality Assurance
IV&V	= Independent Validation and Verification	RADC	= Rome Air Development Center		

Figure 2.3-2 Air Force Acquisition Relationships Involved in Quality Metrics Functions

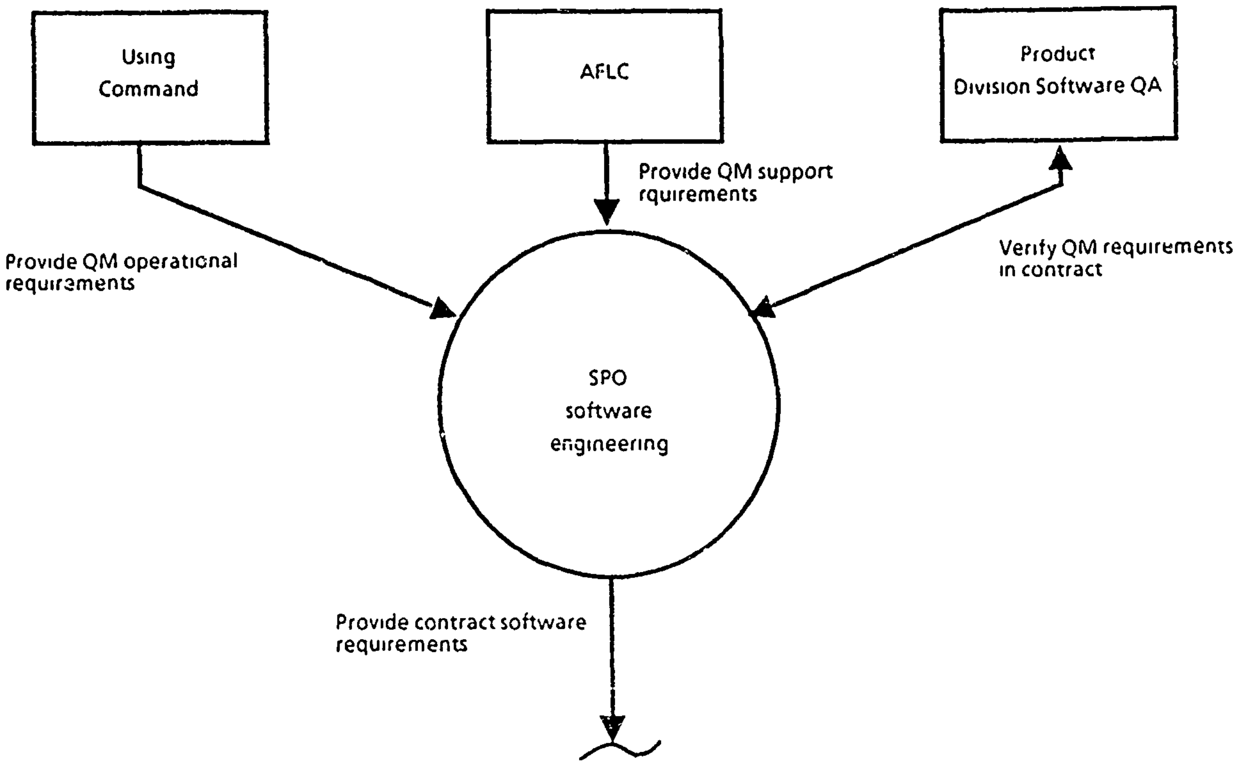


Figure 2.3-3

Recommended Responsibilities and Relationships for the QM Specification Function

Table 2.3-1 Organizational Evaluation

CRITERION / ORGANIZATION	TECHNICAL CAPABILITY	LABOR AVAILABILITY	ECONOMY	DATA AVAILABILITY	SCORE* SUMMARY
SPO ENGINEERING	2	2	1	2	7
PRODUCT DIVISION SOFTWARE QA	3	3	1	3	10
AFPRO	2	2	1	1	6
IV&V	1	1	3	2	7

1 = BEST
2 = MEDIUM
3 = WORST

* Lowest Score is Best (Unweighted)

Several organizations also should be involved in the monitoring function. Among the first activities are identifying and negotiating with the organization that will collect and analyze metric data. If that organization is to be another Air Force agency, such as Air Force Contracts Management Division (AFCMD), then the SPO needs to negotiate the effort through a memorandum of agreement. If the organization is to be an IV&V contractor, then the IV&V contract needs to be negotiated. These negotiations must be completed very early in the program before data collection starts, and SPO Software Engineering must ensure that necessary support is provided.

Several organizations could collect and analyze data, including SPO Software Engineering, the Product Division Software QA, the Air Force Plant Representative Office (AFPRO), and an IV&V contractor. The following criteria were established to aid in selecting an organization: technical capability, labor availability, economy, and data availability. Technical capability refers to the depth of technical understanding of software by people in the organization. Labor availability refers to availability of qualified people to perform this additional task (i.e., currently available or readily obtainable). Economy refers to the least costly method for the SPO to obtain data. Data availability refers to the ability to access the most current contractor documentation and information. Informal lines of communication greatly influence this factor.

We rated four candidate organizations using these criteria, based on our experience. A score of 1 represents the best conditions and a 3 represents the worst for each criterion. A total unweighted score was determined for each organization, with the lowest score representing the best choice. The evaluation scores are shown in Table 2.3-1.

Several assumptions were made for scoring. The first was that all criteria are weighted equally; actually, however, technical capability and labor availability may be overriding factors for selection. For technical capability, it was assumed that Product Division Software QA groups are unlikely to be able to obtain people experienced in both software engineering and QA to perform that job. For economy, it was assumed that any Air Force person (civilian or military) is a free resource for the SPO. Otherwise, the SPO must pay for IV&V contractor services. Data availability scores include the assumption that the IV&V contractor works for SPO Software Engineering

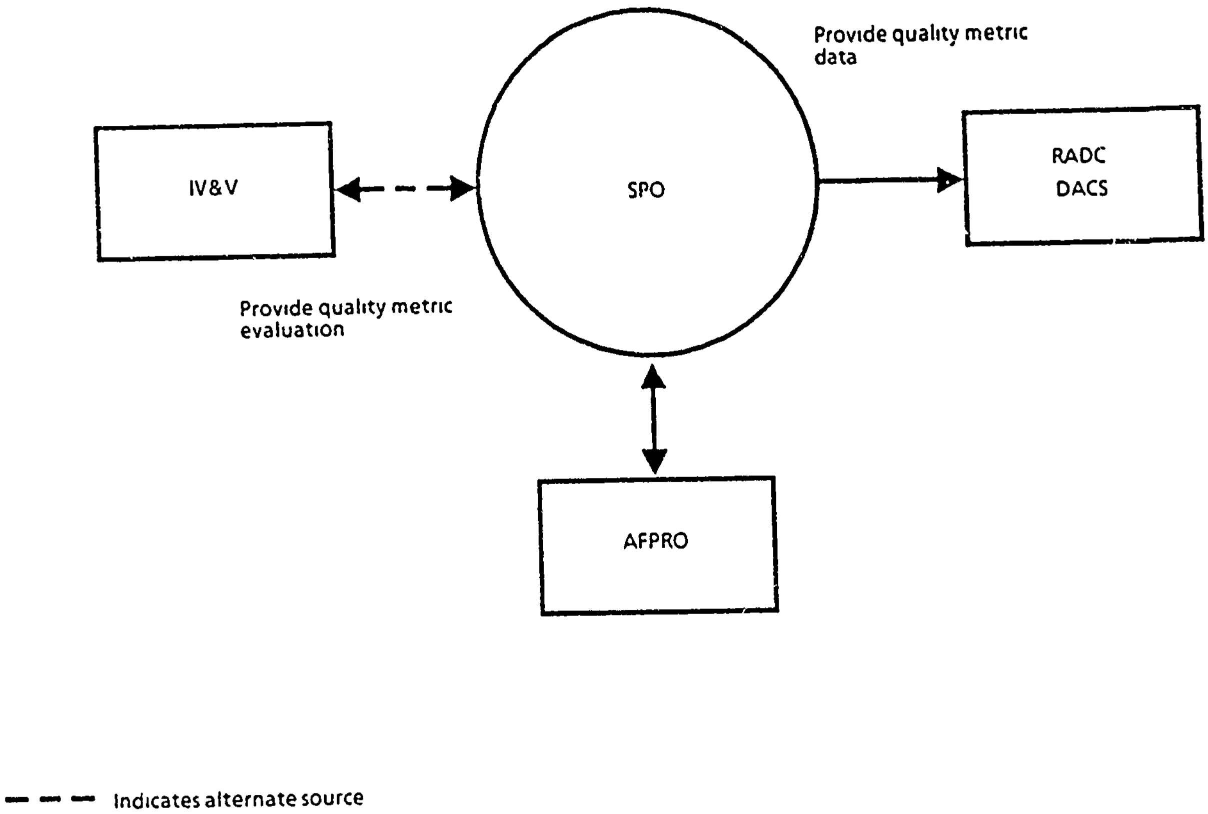


Figure 2.3-4

Recommended Responsibilities and Relationships for the QM Monitoring Function

and that good communication channels are established. These assumptions may not be valid in all situations.

The AFPRO received the lowest score and, therefore, was rated best. It is generally recommended that the AFPRO perform data collection and analysis for the SPO. When this cannot be negotiated, it is recommended that an IV&V contractor be assigned this task. Although SPO Software Engineering and the IV&V contractor are rated equally, the recommendation to use an IV&V contractor was made because of better labor availability. It is recommended that a chart similar to the one shown in Table 2.3-1 be developed early in a program.

A proposed DID, Software Quality Evaluation Report, is contained in Appendix C and can be used to report data collection and analysis results to the software acquisition manager. This feedback enables the manager to track progress, ensure that requirements are being satisfied, and take corrective action when necessary. Recommendations for responsible organizations and relationships for monitoring are shown in Figure 2.3-4. We recommend that the Data and Analysis Center for Software (DACS) at Rome be used as the data base for quality metrics information and that the SPO provide a copy of the quality requirements and all metric data to DACS (e.g., provide a copy of the Software Quality Evaluation Report). This has the advantages of providing one centralized location for all QM data and enabling access to all historical data by any one product division. It also enables large-scale data analysis and correlation to be performed on data from all product divisions. Any changes in QM technology such as new factors, metrics, and worksheet formats should be disseminated from a central point. This concept is illustrated in Figure 2.3-5.

The preceding paragraphs discuss government monitoring only, and the development contractor was not mentioned. Because quality factor requirements are included as contractual requirements, the development contractors must also monitor achieved quality levels to show compliance. However, to ensure that data and reports received by the SPO are unbiased, we recommend that the government independently monitor achieved quality levels.

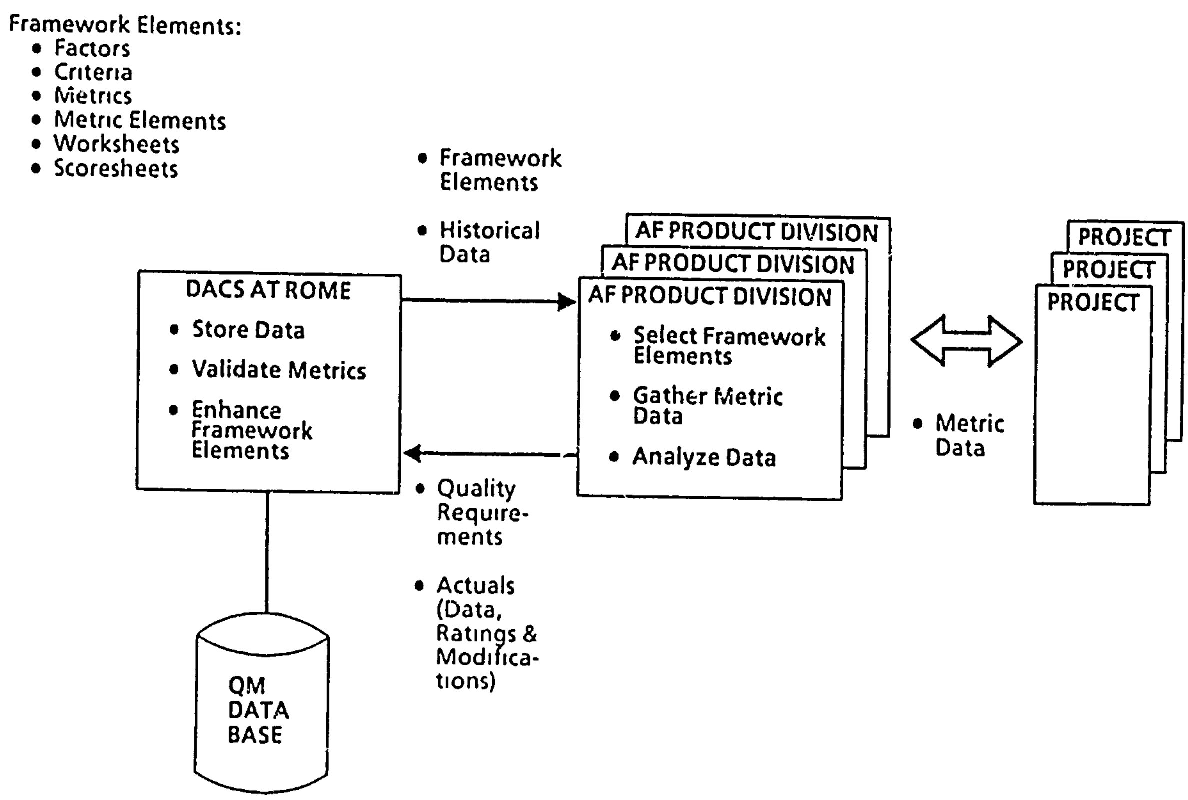


Figure 2.3-5 Relationship between Product Divisions and DACS

2.4 POTENTIAL BENEFITS AND PROBLEMS

This section discusses the potential benefits and problems associated with integrating QM technology into the software acquisition management process and of using QM technology during acquisition.

2.4.1 Benefits

Possible benefits of using QM technology include a higher quality end product, greater emphasis on quality throughout the life cycle, better management control, and life-cycle cost savings. A high-quality end product is possible because required quality levels are specified quantitatively. There is little room for misinterpretation or for undesirable results such as a highly efficient but unreliable and unmaintainable product. The acquisition manager is assured that the end product is of the required degree of quality. Also, other software requirements are considered at the same time that quality requirements are being specified. This means that the quality requirements should be reasonable and should not conflict with functional and performance requirements (or vice versa), thereby increasing the likelihood that all software requirements can be satisfied within allocated resources. In addition, achieved quality levels are monitored throughout development providing increased visibility for control of quality. Periodic application of metrics provides the acquisition manager with adequate feedback about software development progress and enables early redirection if necessary. Finally, evaluating specific low metric scores provides an additional mechanism for detecting deficiencies and anomalies in requirements, design, and code.

Life-cycle cost savings are possible for several reasons. Using metrics to detect deficiencies and anomalies enables correction during development. Correction at this time is less costly than during operation and maintenance. Also, it is possible to be more precise about funding for quality. If adequate quality levels are achieved during development, it is unnecessary to spend more effort in raising quality levels or in developing a near-perfect product.

The greatest cost savings potential comes from having certain qualities actually built into the software. For example, if system A has a high level of reusability built into

the software, then cost savings result from building system B reusing a portion of system A software. These potential cost savings are available for other quality factors such as flexibility, portability, interoperability, and expandability. Details for considering cost are described in Section 4.0 of the specification guidebook (Vol. II).

Other benefits can also be realized. For example, use of QM technology can provide the acquisition manager an added assurance that the required degree of reliability is achieved in the final product. This would be especially important in acquisitions involving space applications or nuclear armaments.

2.4.2 Problems

There are potential technical and administrative problems when using quality metrics in acquisitions; i.e., in integrating QM technology into the Air Force software acquisition process. Problems could arise during one of the most important tasks, that of maintaining a current QM technology baseline. Baseline changes could result from, for example, changes in quality factor ratings, new factor ratings being established, new metrics being established, and metrics being validated for new application areas. Changes could originate from any product division using QM technology. Using DACS would minimize the risk of such problems as: multiple baselines in the Product Divisions, duplication of validation efforts, and use of outdated information (e.g., outdated ratings).

A potential problem could arise where subjective judgment is required in scoring some metrics. Two people gathering metric data from the same software products could score the worksheets differently. This risk has been minimized by rewriting the questions on the metric worksheets so that they are clear, simple, and understandable. Also, metric element explanations have been included for clarification. As more historical information becomes available, it will be possible to do a reasonableness check on worksheet data entries, based on previous data ranges. However, we recommend that experienced personnel perform data collection and that education and training be provided for personnel involved with QM technology.

Another potential problem might arise when attempting to automate portions of the data gathering task through an automated measuring tool. This type of tool scans source code and outputs statistics on the code (e.g., percentage of comments, number of specific constructs). The scanner is language dependent and must be developed for each language, but standardization on a language (e.g., Ada) will minimize cost.

Problems with organizational structures and manpower may be encountered when implementing QM technology at the product divisions. Program offices do not have QA divisions. QA in the program office is usually done by Engineering. In addition, software QA organizations in the product divisions are relatively new. These organizations are trying to define their role in the acquisition process and their relationship to the program offices. Absence of a well-defined organizational structure for software QA could lead to disagreements over assigning QM responsibilities. Either organization could resist accepting responsibility for QM functions because of staffing problems. Program offices are usually not fully staffed with software engineers; to accept more responsibilities without additional personnel would be difficult. Software QA organizations have small staffs and find it difficult to hire qualified personnel. A person with experience in both software engineering and QA is required, but few software engineers are interested in QA assignments. Staffing problems should receive attention during implementation of QM technology in the Air Force software acquisition process.

3.0 QUALITY METRICS FRAMEWORK

This section describes elements of the software quality framework. Terminology and concepts introduced in this section are used throughout subsequent sections.

The goals of quality metrics (QM) technology are to enable a software acquisition manager to (1) specify the types and degrees of software qualities desired in the end product and (2) predict end-product quality levels through measuring the degree of those qualities present during development. The Rome Air Development Center (RADC) quality program (see Sec. 1.1) has established a model for viewing software quality. Figure 2.2-2 depicts this model, showing a hierarchical relationship between a quality factor, criteria, and metrics. Criteria and metrics are factor attributes.

Quality factors (e.g., reliability, usability, correctness, and maintainability) are user-oriented terms, each representing an aspect of software quality. Thirteen quality factors are used to specify the types of qualities wanted in a particular software product. Product environment and expected use affect emphasis. For example, if human lives could be affected, integrity, reliability, correctness, verifiability, and survivability would be emphasized. If the software is expected to have a long life cycle, maintainability and expandability would be emphasized.

Criteria are software-oriented terms representing software characteristics. For example, operability and training are criteria for usability. The degree to which these characteristics are present in the software is an indication of the degree of presence of an aspect of quality (i.e., a quality factor).

Metrics are software-oriented details of a characteristic (a criterion) of the software. Each metric is defined by a number of metric elements. The metric elements enable quantification of the degree of presence of criteria and, hence, factors. "Are all the errors specified which are to be reported to the operator/user?" is an example metric element question for the criterion operability (see worksheet 0, OP.1(2), App. A).

Using the methodology described in Volume II, Section 4.0, the acquisition manager is responsible for specifying needed quality factors by priority, with quality levels

Table 3.1-1 Software Quality Factor Definitions and Rating Formulas

ACQUISITION CONCERN	QUALITY FACTOR	DEFINITION	RATING FORMULA
PERFORMANCE	EFFICIENCY	RELATIVE EXTENT TO WHICH A RESOURCE IS UTILIZED (i.e. STORAGE SPACE, PROCESSING TIME, COMMUNICATION TIME)	$1 - \frac{\text{ACTUAL RESOURCE UTILIZATION}}{\text{ALLOCATED RESOURCE UTILIZATION}}$
	INTEGRITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM WITHOUT FAILURES DUE TO UNAUTHORIZED ACCESS TO THE CODE OR DATA WITHIN A SPECIFIED TIME PERIOD	$1 - \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	RELIABILITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM WITHOUT ANY FAILURES WITHIN A SPECIFIED TIME PERIOD	$1 \ \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	SURVIVABILITY	EXTENT TO WHICH THE SOFTWARE WILL PERFORM AND SUPPORT CRITICAL FUNCTIONS WITHOUT FAILURES WITHIN A SPECIFIED TIME PERIOD WHEN A PORTION OF THE SYSTEM IS INOPERABLE	$1 \ \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	USABILITY	RELATIVE EFFORT FOR USING SOFTWARE (TRAINING AND OPERATION) (e.g. FAMILIARIZATION, INPUT PREPARATION, EXECUTION, OUTPUT INTERPRETATION)	$1 \ \frac{\text{LABOR DAYS TO USE}}{\text{LABOR YEARS TO DEVELOP}}$
DESIGN	CORRECTNESS	EXTENT TO WHICH THE SOFTWARE CONFORMS TO ITS SPECIFICATIONS AND STANDARDS	$1 \ \frac{\text{ERRORS}}{\text{LINES OF CODE}}$
	MAINTAINABILITY	EASE OF EFFORT FOR LOCATING AND FIXING A SOFTWARE FAILURE WITHIN A SPECIFIED TIME PERIOD	1 0.1 (AVERAGE LABOR DAYS TO FIX)
	VERIFIABILITY	RELATIVE EFFORT TO VERIFY THE SPECIFIED SOFTWARE OPERATION AND PERFORMANCE	$1 \ \frac{\text{EFFORT TO VERIFY}}{\text{EFFORT TO DEVELOP}}$
ADAPTATION	EXPANDABILITY	RELATIVE EFFORT TO INCREASE THE SOFTWARE CAPABILITY OR PERFORMANCE BY ENHANCING CURRENT FUNCTIONS OR BY ADDING NEW FUNCTIONS OR DATA	$1 \ \frac{\text{EFFORT TO EXPAND}}{\text{EFFORT TO DEVELOP}}$
	FLEXIBILITY	EASE OF EFFORT FOR CHANGING THE SOFTWARE MISSIONS, FUNCTIONS, OR DATA TO SATISFY OTHER REQUIREMENTS	1- 0.05 (AVERAGE LABOR DAYS TO CHANGE)
	INTEROPERABILITY	RELATIVE EFFORT TO COUPLE THE SOFTWARE OF ONE SYSTEM TO THE SOFTWARE OF ANOTHER SYSTEM	$1 \ \frac{\text{EFFORT TO COUPLE}}{\text{EFFORT TO DEVELOP}}$
	PORTABILITY	RELATIVE EFFORT TO TRANSPORT THE SOFTWARE FOR USE IN ANOTHER ENVIRONMENT (HARDWARE CONFIGURATION AND/OR SOFTWARE SYSTEM ENVIRONMENT)	$1 - \frac{\text{EFFORT TO TRANSPORT}}{\text{EFFORT TO DEVELOP}}$
	REUSABILITY	RELATIVE EFFORT TO CONVERT A SOFTWARE COMPONENT FOR USE IN ANOTHER APPLICATION	$1 - \frac{\text{EFFORT TO CONVERT}}{\text{EFFORT TO DEVELOP}}$

NOTE: THE RATING VALUE RANGE IS FROM 0 TO 1. IF THE VALUE IS LESS THAN 0, THE RATING VALUE IS ASSIGNED TO 0.

commensurate with cost consideration. Factor requirements are provided as part of the software requirements (along with operational, performance, and design requirements). This enables the corresponding criteria and metrics to be identified and used to measure the degree of presence of desired qualities at key review points during development, allowing periodic predictions of the quality level for the final product. Metric worksheets and scoresheets help in applying the metrics and in determining metric scores.

3.1 SOFTWARE QUALITY FACTORS

Thirteen software quality factors are identified in Table 2.2-1, with the user concern that characterizes the need for each type of quality. Quality factors are shown grouped under one of three acquisition concerns: performance, design, or adaptation. An acquisition manager specifying requirements for software will likely do so in a DOD-STD-SDS format in four main areas: (1) software performance characteristics (performance), (2) software design and construction (design), (3) anticipated software expansion or reuse (adaptation), and (4) quality assurance (including quality metrics). The similarity of areas and acquisition concerns enables the acquisition manager to easily identify and select quality factor categories and specific factors of interest. Quality criteria are similarly categorized (see Sec. 3.2); thus, selecting criteria and metrics is simplified.

3.1.1 Factor Definitions and Rating Formulas

Quality factor definitions and factor rating formulas are shown in Table 3.1-1. Rating formulas quantify user concerns for the final product. The formulas use three types of measurements: (1) number of errors per lines of code (2) effort to perform an action and (3) utilization of resources. Ratings should fall in the range from zero to one. The rating formula for reliability is one minus the number of errors per lines of code. For example, if one error per 1,000 lines of code occur during a given time period (e.g., during operational testing and evaluation) the rating formula shows a reliability level of 0.999 (1-1/1,000 = 0.999).

During software development, metrics are applied to software products, and a metric score is calculated for the appropriate factors. This metric score is an estimation (or

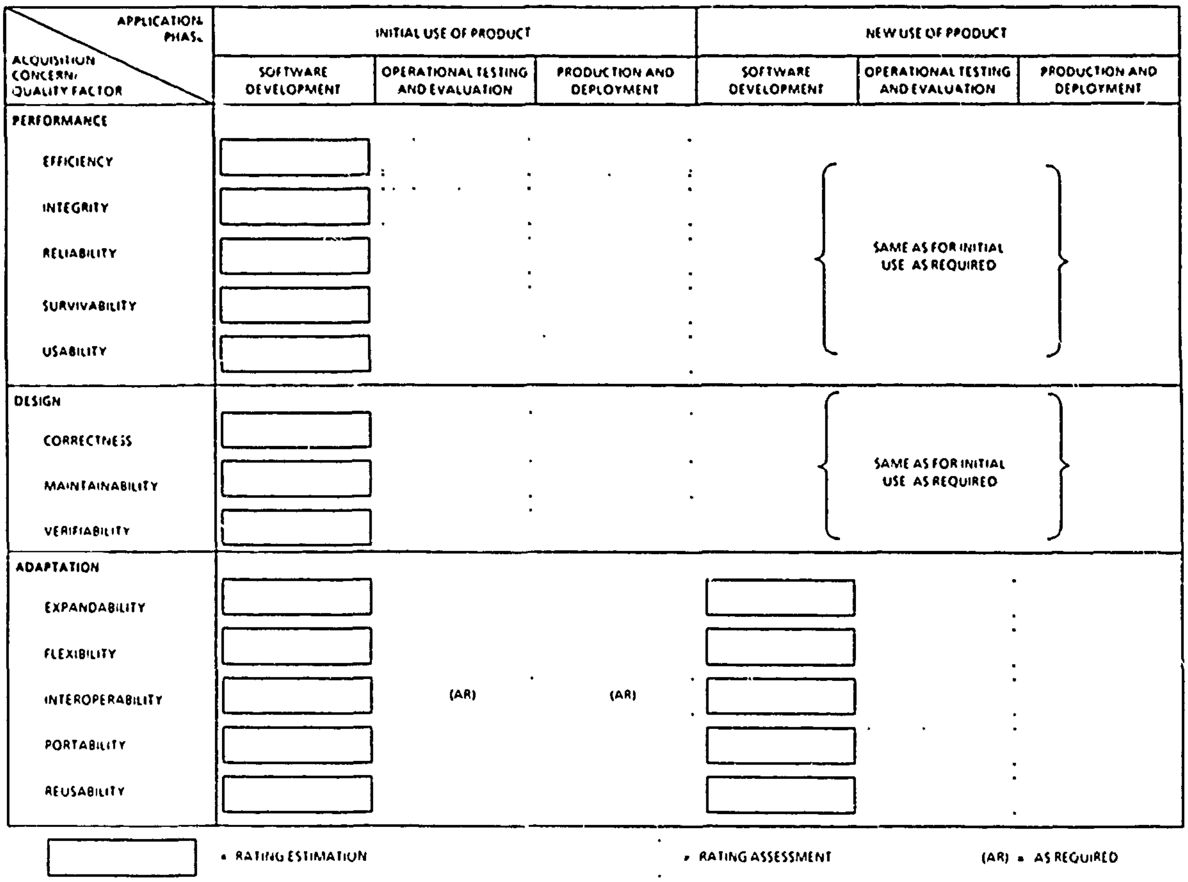


Figure 3.1-1 Rating Estimation and Rating Assessment Windows

prediction) of what the quality level will be for the final product. Figure 3.1-1 indicates the timeframes during which rating values are estimated through metric scores (closed box) and the timeframes during which rating values can be assessed by using actual data and the rating formula (dotted box). For example, the rating value for reliability is estimated by using metric scores during software development. During operational testing and evaluation and during production and deployment, actual data on number of errors per lines of code become available to assess the rating and evaluate predictions made during development. Exact correlations between metric scores and rating values have not been established. Research has only shown that higher metric scores during development result in higher quality end products. Table 3.1-2 shows a range of values for each rating formula that might occur when using actual data (e.g., during production and deployment) to assess rating values. The values shown are hypothetical.

The following paragraphs describe the factors and rating formulas in each acquisition concern category.

Performance. Performance quality factors deal both with the ability of the software to function and with error occurrences that affect software functioning. Low quality levels predict poor software performance. These quality factors are efficiency, integrity, reliability, survivability, and usability.

<u>Efficiency</u> deals with utilization of a resource. The rating formula for efficiency is in terms of actual utilization of a resource and budgeted allocation for utilization. For example, if a unit is budgeted for 10% available memory and actually uses 7%, the rating formula shows an efficiency level of 0.3 (1 - 0.07/0.10 = 0.3).

<u>Integrity</u> deals with software security failures due to unauthorized access. The rating formula for integrity is in terms of number of integrity-related software errors occuring during a given time (e.g., during operational testing and evaluation) and total number of executable lines of source code. This formula is similar to the formula for reliability; the difference is that reliability is concerned with all software errors, and integrity is concerned only with the subset of errors that affect integrity. For example, if three integrity-related errors per 10,000 lines of code occurred during operational testing and evaluation, the rating formula shows an integrity level of 0.9997 (1 - 1/10,000 = 0.9997).

Table 3.1-2 Quality Factor Ratings

Quality factor	Rating formula	Rating information			
Efficiency	$1 - \frac{\text{Actual utilization}}{\text{Allocated utilization}}$	Value	0.1	0.3	0.5
		% utilization	90%	70%	50%
Integrity	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10,000	3/10,000	1/10 000
Reliability	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.995	0.997	0.999
		Errors/LOC	5/1,000	3/1,000	1/1 000
Survivability	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10 000	3/10 000	1/10 000
Usability	$1 - \frac{\text{Labor-days to use}}{\text{Labor-years to develop}}$	Value	0.5	0.7	0.9
		Days/years	5/10	6/20	10/100
Correctness	$1 - \frac{\text{Errors}}{\text{Lines of code}}$	Value	0.9995	0.9997	0.9999
		Errors/LOC	5/10 000	3/10,000	1/10 000
Maintainability	1- 0.1 (average labor-days to fix)	Value	0.8	0.9	0.95
		Average labor-days	2.0	1.0	0.5
Verifiability	$1 - \frac{\text{Effort to verify}}{\text{Effort to develop}}$	Value	0.4	0.5	0.6
		% effort	60%	50%	40%
Expandability	$1 - \frac{\text{Effort to expand}}{\text{Effort to develop}}$	Value	0.8	0.9	0.95
		% effort	20%	10%	5%
Flexibility	1- 0.05 (average labor-days to change)	Value	0.8	0.9	0.95
		Average labor-days	40	20	10
Interoperability	$1 - \frac{\text{Effort to couple}}{\text{Effort to develop}}$	Value	0.9	0.95	0.99
		% effort	10	5	1
Portability	$1 - \frac{\text{Effort to transport}}{\text{Effort to develop}}$	Value	0.9	0.95	0.99
		% effort	10	5	1
Reusability	$1 - \frac{\text{Effort to convert}}{\text{Effort to develop}}$	Value	0.4	0.6	0.8
		% effort	60	40	20

Reliability concerns any software failure. The rating formula for reliability is in terms of total number of software errors occurring during a specified time and total number of executable lines of source code. For example, if three errors per 1,000 lines of code occurred during operational testing and evaluation, the rating formula shows a reliability level of 0.997 (1 - 3/1,000 = 0.997).

The concern with survivability is that software continue to perform (e.g., in a degraded mode) even when a portion of the system has failed. The rating formula for survivability is in terms of number of survivability-related errors (the subset of errors that affect survivability) occurring during a specified time and total number of executable lines of source code. This formula is similar to the formula for reliability.

Usability deals with relative effort involved in learning about and using software. The rating formula for usability is in terms of average effort to use software (to train for using it and to operate it) and original development effort. This formula considers size of the software system in rating usability. It is recommended that effort to use be expressed in labor-days and effort for original development be expressed in labor-years to maintain a scoring range consistent with that of other factors. For example, if 10 labor-days were required for training on a system that required 100 labor-years to develop, the rating formula shows a usability level of 0.9 (1 - 10/100 = 0.9); and if five labor-days were required for training on a system that required 10 labor-years to develop, the rating formula shows a usability level of 0.5 (1 -5/10 = 0.5).

Design. Design quality factors deal mainly with software failure and correction. Low quality levels usually result in repeating a portion of the development process (e.g., redesign, recode, reverify); hence the term design. The factors are correctness, maintainability, and verifiability.

Correctness deals with the extent to which software design and implementation conform to specifications and standards. Criteria of correctness (completeness, consistency, and traceability) deal exclusively with design and documentation formats. Under the three criteria there are no metrics dealing with content material affecting software operation or performance. The rating formula for correctness is in terms of number of specifications-related and standards-related errors that occur after formal

release of the specifications and standards and total number of executable lines of source code. This formula is also similar to the formula for reliability; the difference is that correctness is concerned only with that subset of errors related to violations of specified requirements and nonconformance to standards.

Maintainability is concerned with ease of effort in locating and fixing software failures. The rating formula for maintainability is in terms of average number of labor-days to locate and fix an error within a specified time (e.g., during production and deployment). For example, if an average of 0.5 labor-days were required to locate and fix errors during production and deployment, the rating formula shows a maintainability level of 0.95 (1 - (0.1 x 0.5) = 0.95).

Verifiability deals with software design characteristics affecting the effort to verify software operation and performance. The rating formula for verifiability is in terms of effort to verify software operation and performance and original development effort. This formula is similar to the adaptation, effort-ratio formulas. For example, if 40% of the development effort is spent reviewing and testing software, the rating formula shows a verifiability level of 0.6 (1 - 0.40/1.00 = 0.6).

Adaptation. These quality factors deal mainly with using software beyond its original requirements, such as extending or expanding capabilities and adapting for use in another application or in a new environment. Low quality levels predict relatively high costs for new software use. Quality factors are expandability, flexibility, interoperability, portability, and reusability.

Expandability deals with relative effort in increasing software capabilities or performance. The rating formula for expandability is in terms of effort to increase software capability and performance and original development effort. For example, if five labor-months were spent enhancing software performance for software that orignally took 100 labor-months to develop, the rating formula shows an expandability level of 0.95 (1 - 5/100 = 0.95).

Flexibility deals with ease of effort in changing software to accommodate changes in requirements. The rating formula for flexibility is in terms of average effort to change software to satisfy other (i.e., new or modified) requirements within a

specified time. For example, if an average of one labor-day was required to modify software functioning during operational testing and evaluation, the rating formula shows a flexibility level of 0.95 (1 - (0.05 x 1) = 0.95).

Interoperability is concerned with relative effort in coupling software of one system to software of one or more other systems. The rating formula for interoperability is in terms of effort to couple and original development effort and is similar to the formula for expandability.

Portability deals with relative effort involved in transporting software to another environment (e.g., different host processor, operating system, executive). The rating formula for portability is in terms of effort to transport software for use in another environment and original development effort and is similar to the formula for expandability.

Reusability is concerned with relative effort for converting a portion of software for use in another application. The rating formula for reusability is in terms of effort to convert software for use in another application and original development effort and is similar to the formula for expandability.

If adaptation effort is greater than original development effort, the effort-ratio formulas will yield a quality level value less than zero. In this case, the quality level value is assigned to zero. (This situation is considered unlikely because it would probably be less expensive to develop a new product than to adapt an existing one.)

3.1.2 Quality Factor Interrelationships

Relationships exist among quality factors; some relationships are synergistic and others conflicting. Specifying requirements for more than one type of quality for a product can possibly have either a beneficial or an adverse effect on cost to provide the quality. Factor relationships and relative cost to provide are discussed in Section 4.0 of the specification guidebook (Vol. II).

Table 3.2-1 Software Quality Factors and Criteria

ACQUISITION CONCERN	CRITERION/ACRONYM		PERFORMANCE					DESIGN			ADAPTATION				
	FACTOR/ACRONYM		EFFICIENCY EF	INTEGRITY IG	RELIABILITY RL	SURVIVABILITY SV	USABILITY US	CORRECTNESS CR	MAINTAINABILITY MA	VERIFIABILITY VE	EXPANDABILITY EX	FLEXIBILITY FX	INTEROPERABILITY IP	PORTABILITY PO	REUSABILITY RU
PERFORMANCE	ACCURACY	AC			X										
	ANOMALY MANAGEMENT	AM			X	X									
	AUTONOMY	AU				X									
	DISTRIBUTEDNESS	DI				X									
	EFFECTIVENESS COMMUNICATION	EC	X												
	EFFECTIVENESS - PROCESSINC	EP	X												
	EFFECTIVENESS - STORAGE	ES	X												
	OPERABILITY	OP					X								
	RECONFIGURABILITY	RE				X									
	SYSTEM ACCESSIBILITY	SS		X											
	TRAINING	TN					X								
DESIGN	COMPLETENESS	CP						X							
	CONSISTENCY	CS						X	X						
	TRACEABILITY	TC						X							
	VISIBILITY	VS							X	X					
ADAPTATION	APPLICATION INDEPENDENCE	AP													X
	AUGMENTABILITY	AT									X				
	COMMONALITY	CL											X		
	DOCUMENT ACCESSIBILITY	DO													X
	FUNCTIONAL OVERLAP	FO											X		
	FUNCTIONAL SCOPE	FS													X
	GENERALITY	CE									X	X			X
	INDEPENDENCE	ID											X	X	X
	SYSTEM CLARITY	ST													X
	SYSTEM COMPATIBILITY	SY											X		
	VIRTUALITY	VR									X				
GENERAL	MODULARITY	MO				X			X	X	X	X	X	X	X
	SELF DESCRIPTIVENESS	SD							X	X	X	X		X	X
	SIMPLICITY	SI			X				X	X	X	X			X

3.2 SOFTWARE QUALITY CRITERIA

Criteria are software-oriented terms representing software characteristics. Software quality criteria can be grouped under the same three aquisition concerns as quality factors: performance, design, and adaptation. Table 3.2-1 shows the relationship of criteria to quality factors. Four categories for criteria are shown: performance, design, adaptation, and general. Each criterion is an attribute of one or more quality factors. The criteria in the first three categories are solely attributes of factors within the same acquisition concern (i.e., performance, design, and adaptation). Criteria in the fourth category are factor attributes within more than one acquisition concern.

Criteria and factors within each category are listed alphabetically for easy referencing. Alphabetizing by name or by acronym gives the same sequence. Criteria definitions are listed in Table 3.2-2.

3.3 SOFTWARE QUALITY METRICS

Metrics are software-oriented details of a software characteristic (a criterion). Each criterion consists of one or more metrics. Each metric is an attribute of only one criterion. Table 3.3-1 lists the name and acronym of each criterion (in alphabetical order) and the name and acronym of each metric that is an attribute of that criterion. Metric acronyms are acronym extensions of the parent criterion. For example, the acronym for the criterion commonality is CL; the acronym for the three metric attributes are CL.1, CL.2, and CL.3.

Each metric is defined by one or more metric elements. Metric elements are detailed questions applied to software products; answers to them enable quantification of metrics and of the parent criterion and factor. Metric elements are designated by acronym only (no name) and are listed on the metric worksheets. Acronym designation is an extension of the parent metric acronym. For example, the 14 metric element acronyms for the metric CL.1 are CL.1 (1) through CL.1 (14).

Table 3.2-2 Quality Criteria Definitions

ACQUISITION CONCERN	CRITERION	ACRONYM	DEFINITION
PERFORMANCE	ACCURACY	AC	• Those characteristics of software which provide the required precision in calculations and outputs
	ANOMALY MANAGEMENT	AM	• Those characteristics of software which provide for continuity of operations under and recovery from non-nominal conditions
	AUTONOMY	AU	• Those characteristics of software which determine its non-dependency on interfaces and functions
	DISTRIBUTEDNESS	DI	• Those characteristics of software which detemrine the degree to which software functions are geographically or logically separated within the system
	EFFECTIVENESS-COMM	EC	• Those characteristics of the software which provide for minimum utilization of communications resources in performing functions
	EFFECTIVENESS-PROCESSING	EP	• Those characteristics of the software which provide for minimum utilization of processing resources in performing functions
	EFFECTIVENESS STORAGE	ES	• Those characteristics of the software which provide for minimum utilization of storage resources
	OPERABILITY	OP	• Those characteristics of software which determine operations and procedures concerned with operation of software and which provide useful inputs and outputs which can be assimilated
	RECONFIGURABILITY	RE	• Those characteristics of software which provide for continuity of system operation when one or more processors, storage units, or communication links fails
	SYSTEM ACCESSIBILITY	SS	• Those characteristics of software which provide for control and audit of access to the software and data
	TRAINING	TN	• Those characteristics of software which provide transition from current operation and provide initial familiarization
DESIGN	COMPLETENESS	CP	• Those characteristics of software which provide full implementation of the functions required
	CONSISTENCY	CS	• Those characteristics of software which provide for uniform design and implementation techniques and notation
	TRACEABILITY	TC	• Those characteristics of software which provide a thread of origin from the implementation to the requirements with respect to the specified development envelope and operational environment
	VISIBILITY	VS	• Those characteristics of software which provide status monitoring of the development and operation
ADAPTATION	APPLICATION INDEPENDENCE	AP	• Those characteristics of software which detemrine its nondependency on database system microcode, computer architecture, and algorithms
	AUGMENTABILITY	AT	• Those characteristics of software which provide for expansion of capability for functions and data
	COMMONALITY	CL	• Those characteristics of software which provide for the use of interface standards for protocols, routines and data representations
	DOCUMENT ACCESSIBILITY	DO	• Those characteristics of software which provides for easy access to software and selective use of its components
	FUNCTIONAL OVERLAP	FO	• Those characteristics of software which provide common functions to both systems
	FUNCTIONAL SCOPE	FS	• Those characteristics of software which provide commonality of functions among applications
	GENERALITY	FE	• Those characteristics of software which provide breadth to the functions performed with respect to the application
	INDEPENDENCE	ID	• Those characteristics of software which determine its non-dependency on software environment (computing system operating system utilities, input, output routines, libraries)
	SYSTEM CLARITY	ST	• Those characteristics of software which provide for clear description of program structure in a non-complex and understandable manner
	SYSTEM COMPATIBILITY	SY	• Those characteristics of software which provide the hardware, software, and communication compatibility of two systems
	VIRTUALITY	VR	• Those characteristics of software which present a system that does not require user knowledge of the physical, logical or topological characteristics
GENERAL	MODULARITY	MO	• Those characteristics of software which provide a structure of highly cohesive components with optimum coupling
	SELF-DESCRIPTIVENESS	SD	• Those characteristics of software which provide explanation of the implementation of functions
	SIMPLICITY	SI	• Those characteristics of software which provide for definition and implementation of functions in the most noncomplex and understandable manner

Table 3.3-1 Quality Metrics Summary

CRITERION		METRIC	
NAME	ACRONYM	NAME	ACRONYM
ACCURACY	AC	ACCURACY CHECKLIST	AC.1
ANOMALY MANAGEMENT	AM	ERROR TOLERANCE/CONTROL IMPROPER INPUT DATA COMPUTATIONAL FAILURES HARDWARE FAULTS DEVICE ERRORS COMMUNICATIONS ERRORS NODE/COMMUNICATION FAILURES	AM.1 AM.2 AM.3 AM 4 AM 5 AM 6 AM.7
APPLICATION INDEPENDENCE	AP	DATA BASE MANAGEMENT IMPLEMENTATION INDEPENDENCE DATA STRUCTURE ARCHITECTURE STANDARDIZATION MICROCODE INDEPENDENCE FUNCTIONAL INDEPENDENCE	AP 1 AP 2 AP 3 AP 4 AP 5
AUGMENTABILITY	AT	DATA STORAGE EXPANSION COMPUTATION EXTENSIBILITY CHANNEL EXTENSIBILITY DESIGN EXTENSIBILITY	AT.1 AT 2 AT.3 AT 4
AUTONOMY	AU	INTERFACE COMPLEXITY SELF-SUFFICIENCY	AU.1 AU.2
COMMONALITY	CL	COMMUNICATIONS COMMONALITY DATA COMMONALITY COMMON VOCABULARY	CL.1 CL.2 CL.3
COMPLETENESS	CP	COMPLETENESS CHECKLIST	CP 1
CONSISTENCY	CS	PROCEDURE CONSISTENCY DATA CONSISTENCY	CS 1 CS 2
DISTRIBUTEDNESS	DI	DESIGN STRUCTURE	DI 1
DOCUMENT ACCESSIBILITY	DO	ACCESS TO DOCUMENTATION WELL-STRUCTURED DOCUMENTATION	DO 1 DO 2
EFFECTIVENESS-COMMUNICATION	EC	COMMUNICATION EFFECTIVENESS MEASURE	EC.1
EFFECTIVENESS-PROCESSING	EP	PROCESSING EFFECTIVENESS MEASURE DATA USAGE EFFECTIVENESS MEASURE	EP.1 EP 2
EFFECTIVENESS-STORAGE	ES	STORAGE EFFECTIVENESS MEASURE	ES.1
FUNCTIONAL OVERLAP	FO	FUNCTIONAL OVERLAP CHECKLIST	FO.1
FUNCTIONAL SCOPE	FS	FUNCTION SPECIFICITY FUNCTION COMMONALITY FUNCTION SELECTIVE USABILITY	FS.1 FS.2 FS.3
GENERALITY	GE	UNIT REFERENCING UNIT IMPLEMENTATION	GE.1 GE.2
INDEPENDENCE	ID	SOFTWARE INDEPENDENCE FROM SYSTEM MACHINE INDEPENDENCE	ID 1 ID 2
MODULARITY	MO	MODULAR IMPLEMENTATION MODULAR DESIGN	MO.1 MO 2
OPERABILITY	OP	OPERABILITY CHECKLIST USER INPUT COMMUNICATIVENESS USER OUTPUT COMMUNICATIVENESS	OP 1 OP 2 OP 3
RECONFIGURABILITY	RE	RESTRUCTURE CHECKLIST	RE.1
SELF-DESCRIPTIVENESS	SD	QUANTITY OF COMMENTS EFFECTIVENESS OF COMMENTS DESCRIPTIVENESS OF LANGUAGE	SD 1 SD 2 SD 3
SIMPLICITY	SI	DESIGN STRUCTURE STRUCTURED LANGUAGE OR PREPROCESSOR DATA AND CONTROL FLOW COMPLEXITY CODING SIMPLICITY SPECIFICITY HALSTEAD'S LEVEL OF DIFFICULTY MEASURE	SI 1 SI 2 SI 3 SI 4 SI 5 SI 6

Table 3.3-1 Quality Metrics Summary (continued)

CRITERION		METRIC	
NAME	ACRONYM	NAME	ACRONYM
SYSTEM ACCESSIBILITY	SS	ACCESS CONTROL ACCESS AUDIT	SS.1 SS.2
SYSTEM CLARITY	ST	INTERFACE COMPLEXITY PROGRAM FLOW COMPLEXITY APPLICATION FUNCTIONAL COMPLEXITY COMMUNICATION COMPLEXITY STRUCTURE CLARITY	ST.1 ST.2 ST.3 ST.4 ST.5
SYSTEM COMPATIBILITY	SY	COMMUNICATION COMPATIBILITY DATA COMPATIBILITY HARDWARE COMPATIBILITY SOFTWARE COMPATIBILITY DOCUMENTATION FOR OTHER SYSTEM	SY.1 SY.2 SY.3 SY.4 SY.5
TRACEABILITY	T	CROSS REFERENCE	TC.1
TRAINING	TN	TRAINING CHECKLIST	TN.1
VIRTUALITY	VR	SYSTEM/DATA INDEPENDENCE	VR.1
VISIBILITY	VS	UNIT TESTING INTEGRATION TESTING CSCI TESTING	VS.1 VS.2 VS.3

3.4 METRIC WORKSHEETS

Metric worksheets are contained in Appendix A. The worksheets contain metric elements as questions. Software products (specifications, documents, and source listings) are used as source information to answer questions on worksheets; answers are then translated into metric element scores (yes = 1, no = 0, and a formula answer results in a score from 0 to 1). This enables scoring of the parent metric, criterion, and factor and results in a quality level indication for the product.

Seven different worksheets are applied in different development phases. Table 3.4-1 indicates the timeframe during an acquisition life-cycle phase when a worksheet is used, shows the software level of abstraction at which the worksheet is applied, and lists key terminology used within the worksheet.

Worksheet 0 is applied to products of system/software requirements analysis. The worksheet is applied at the system level. (For large systems, software may not be a discernible component in the design with separate requirements at the system level. In this case, worksheet 0 is applied at the system segment level.)

Worksheet 1 is applied to products of software requirements analysis. A separate worksheet is used for each CSCI.

Worksheet 2 is applied to products of preliminary design. A separate worksheet is used for each CSCI.

Worksheets 3A and 3B are applied to products of detailed design. A separate worksheet 3A is used for each CSCI. A separate worksheet 3B is used for each unit of a CSCI. Worksheets 3A and 3B are applied together; answers on 3B worksheets for CSCI units are used in scoring the 3A worksheet for that CSCI.

Worksheets 4A and 4B are applied to products of code and unit testing. Worksheets 4A and 4B are applied in the same manner as 3A and 3B. A separate worksheet 4A is used for each CSCI, and a separate worksheet 4B is used for each CSCI unit.

Table 3.4-1 Metric Worksheet/Life-Cycle Correlation

Application Level/ Terminology	Life-Cycle Phase/ Activity	Demonstration & Validation	Full-Scale Development (FSD)						
		System/ Software Requirements Analysis	Software Requirements Analysis	Preliminary Design	Detailed Design	Coding & Unit Testing	CSC Integration & Testing	CSCI - Level Testing	System Integration & Testing
System	• System • System function • CSCI	Metric Worksheet 0					(Selected metric questions are reapplied during the integration and testing phases as indicated in the quality attribute correlation table in Appendix A)		
CSCI	• CSCI • Software function		Metric Worksheet 1						
CSCI	• CSCI • Top-level CSC			Metric Worksheet 2					
CSCI	• CSCI • Top-level CSC • Lower-level CSC • Unit				Metric Worksheet 3A	Metric Worksheet 4A			
UNIT	• Unit				Metric Worksheet 3B	Metric Worksheet 4B			

For the remainder of the development cycle, selected metric questions are reapplied as indicated in the quality attribute correlation table in Appendix A.

Metric worksheets are designed to be applied at specific levels (e.g., CSCI, unit). Worksheets can be applied at other levels; however, some questions may not be applicable. For example, if worksheet 1 were applied to a CSCI function, question CP.1(6) should be deleted or reworded because it only applies at the CSCI level.

Metric worksheets are designed to be applied to software development products identified in DOD-STD-SDS. The minimum product set is listed by software development phase in Table 3.4-2. Each product is identified by title and by DID number. Information from the entire set of products for a particular phase is needed as source material to answer metric questions on the worksheet applicable to that phase. It is not necessary to specify the complete product set for each acquisition, only to have equivalent information available to answer worksheet questions. For example, when acquiring a small system, information regarding the QA plan and software standards may be included as part of the software development plan.

3.5 FACTOR SCORESHEETS

Factor scoresheets are contained in Appendix B. There are 13 factor scoresheets, one for each software quality factor. Scoresheets are used for translating information at the metric element level on the worksheets into a quality level score for a quality factor. Each scoresheet has blanks for the factor and for all attributes of that factor (i.e., criteria, metrics, and metric elements). Worksheet information is transferred to the scoresheets at the metric element level. "Yes" answers are scored as 1; "no" answers are scored as 0; and numeric answers resulting from formulas are transferred directly to scoresheets (scoring range from 0 to 1). Scores are then calculated for the parent metrics, criteria, and factor according to the hierarchical (attribute) relationship indicated on the scoresheet.

Table 3.4-2 Software Development Products

Phase/Product Title	Applicable DID
System/Software Requirements Analysis	
System/Segment Specification	DI-S-X101
Software Development Plan	DI-A-X103
Preliminary Software Requirements Specification	DI-E-X107
Operational Concept Document	DI-M-X125
Software Quality Assurance Plan	DI-R-X105
Software Problem/Change Report	DI-E-X106
Software Standards and Procedures Manual	DI-M-X109
Preliminary Interface Requirements Specification	DI-E-X108
Software Requirements Analysis	
Software Requirements Specification	DI-E-X107
Interface Requirements Specification	DI-E-X108
Preliminary Design	
Software Top-Level Design Document	DI-E-X110
Software Test Plan	DI-T-X116
Preliminary Software User's Manual	DI-M-X121
Preliminary Computer System Operator's Manual	DI-M-X120
Detailed Design	
Software Detailed Design Document	DI-E-X111
Software Test Description	DI-T-X117
Data Base Design Document	DI-E-X113
Interface Design Document	DI-E-X112
Coding and Unit Testing	
Source Code/Listings	(Appendix)
Preliminary Software Test Procedure	DI-T-X118
CSC Integration and Testing	
Software Test Procedure	DI-T-X118
CSCI-Level Testing	
Software Product Specification	DI-E-X114
Software Test Report(s)	DI-T-X119
Software User's Manual	DI-M-X121
Computer System Operator's Manual	DI-M-X120
System Integration and Testing	
Software Product Specification	DI-E-X114
Software Test Report(s)	DI-T-X119
Software User's Manual	DI-M-X121
Computer System Operator's Manual	DI-M-X120

4.0 SOFTWARE QUALITY EVALUATION METHODOLOGY

This section describes a methodology for evaluating achieved software quality for products of the development process. The methodology includes procedures for determining quality level scores and interpreting scoring results.

Methodology Overview. Evaluating software quality is part of a larger process for using quality metrics in software acquisition management. Figure 4.0-1 shows this process in two major parts: software quality specification, including assessment of compliance with requirements, and software quality evaluation (measurement of achieved quality levels). This document, the Software Quality Evaluation Guidebook, provides guidance for evaluation. The Software Quality Specification Guidebook provides guidance for specification.

In Section 2.0, two quality metrics functions—specification and monitoring—were described. Specification includes identifying and detailing quality requirements and monitoring includes gathering and reducing data, comparing results with requirements, and taking corrective action if necessary. Section 4.0 groups these functional activities into two slightly different categories—specification and evaluation—to enable separating the guidebooks for personnel who will be performing different functions. Software quality specification, as shown in Figure 4.0-1, includes identifying and specifying requirements and assessing compliance with these requirements since these are the responsibility of System Program Office (SPO) personnel. Results of compliance assessment are used to initiate corrective action. Software quality evaluation includes only data collection and analysis and generation of the Software Quality Evaluation Report since these are the responsibility of the development contractor or an independent verification and validation (IV&V) contractor (or an Air Force organization, as is discussed in Sec. 2.3).

The process begins early in the system life cycle—usually during system demonstration and validation. We assume that a description of the nature of the system and system needs or requirements exists. This description could be a statement of work or a draft system specification and is the primary basis for identifying software quality factor requirements. A series of procedural steps is performed to determine specific

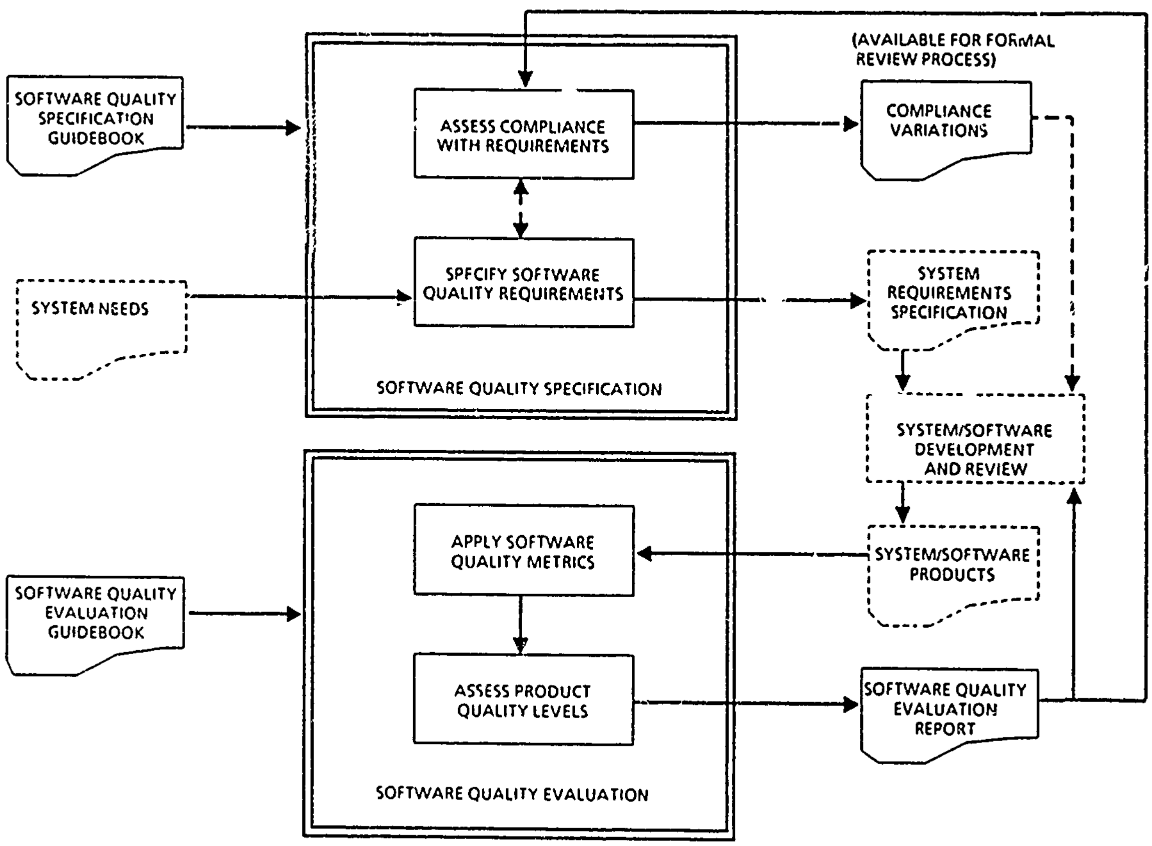


Figure 4.0-1 Software Quality Specification and Evaluation Process

software quality needs and to specify quality requirements. Steps include polling groups such as the Air Force using command and the Air Force Logistics Command (AFLC) in order to provide a comprehensive set of operational and support quality requirements from a quality factor point of view. These steps could be performed by the SPO or the development contractor or through awarding a separate contract.

Software quality requirements are entered into the system requirements specification and are treated as contractual obligations (just the same as technical requirements). As the system contractor proceeds with development, quality requirements from the system requirements specification are allocated to lower level specifications and finally assigned to units within the software detailed design document in a manner similar to that for other requirements. This requirements flow is shown in Figure 4.0-2. Each time during the cycle that development products are released (usually at major review points such as system design review (SDR), software specification review (SSR), preliminary design review (PDR), and critical design review (CDR)), quality metrics, in the form of metric worksheets, are applied to the products. Raw data are then used to calculate scores indicating quality level achieved for each quality factor, and these scores are compared to specified requirements.

Application of metrics and scoring of achieved product quality levels are performed by the development contractor to show compliance with quality requirements. It is anticipated that product evaluation will also be performed in parallel by another group such as an IV&V team, the AFPRO, SPO Software Engineering, or Product Division Software Quality Assurance, as is discussed in Section 2.3. Data collection and analysis results are documented in a Software Quality Evaluation Report (see App. C). This report is reviewed separately at major review points. The report is included in the review package released before the review date. The SPO uses these results to assess compliance with quality requirements and (1) approves or disapproves of compliance variations at the review and/or (2) respecifies quality requirements and ensures that changes are reflected in the system requirements specification.

Use of the Methodology and Guidebooks. The methodology and guidebooks were designed primarily for use on projects during which quality requirements are specified early in the life cycle and achieved quality levels are evaluated periodically during development as was depicted in Figure 1.4-1. The methodology and guidebooks can

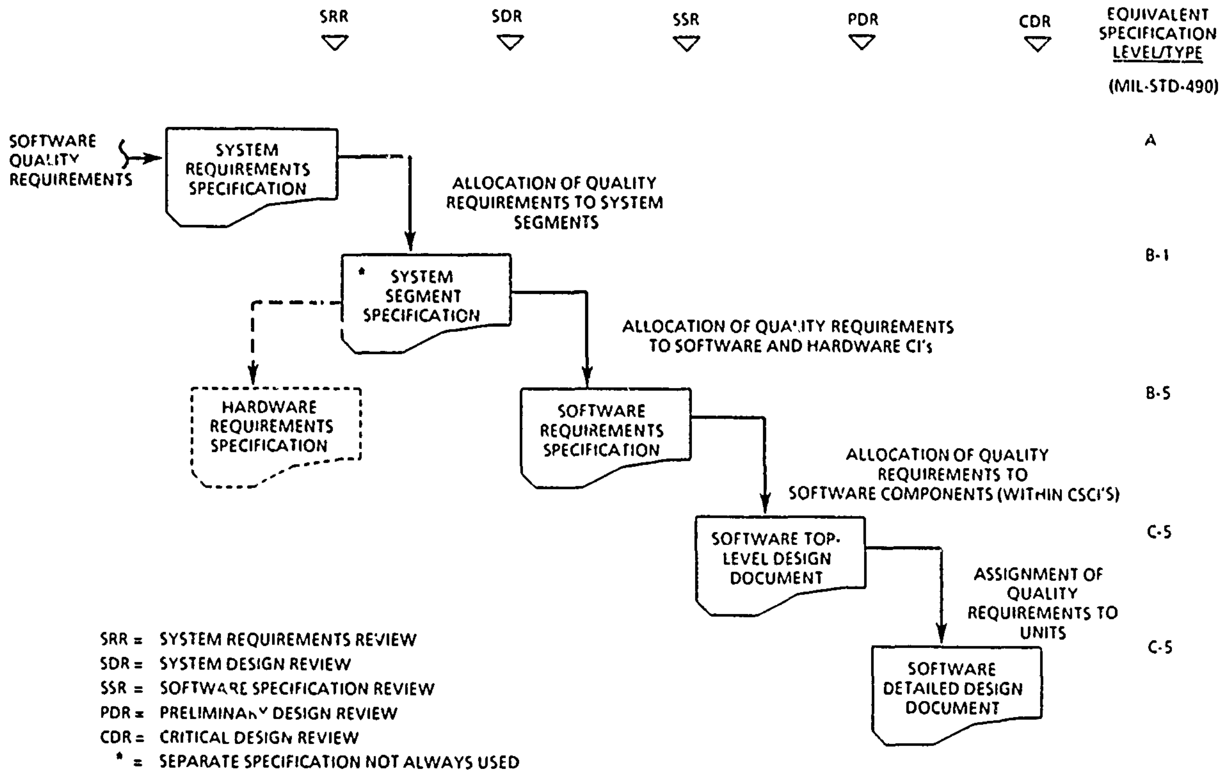


Figure 4.0-2 Flow of Software Quality Requirements

also be used outside the life-cycle context to evaluate particular products such as a specification, design document, source code, or proposal. The purpose might be to evaluate reliability or maintainability of an operational product to determine if it is suitable for an application, to evaluate and compare quality levels of two products for purchasing, or to determine reusability of an operational product as an aid in determining adaptation costs for a new application. The purpose might also be to evaluate quality aspects of new-business proposals or system specifications to help determine a competitive contract award.

The methodology is similar regardless of context. Select important factors, criteria, and metrics. Select appropriate worksheets. Collect data and analyze results. Factor selection should be simplified for applications outside the life-cycle context because it is unlikely that factor cost trades would be performed; however, it is very important that factor interrelationships still be considered to avoid misinterpreting factor scores (explanation in Sec. 4.1.3 of the specification guidebook, Vol. II). Criteria and metrics selections follow factor selection and should consider environmental and application particulars.

Selecting appropriate worksheets requires care to ensure desired results. In using the methodology for a new project with distinct development phases and reviews, a set of products is available at each review point. The metric worksheets are designed to be applied to these products. The products assumed to exist at the end of each software development phase are identified by title and data item description (DID) number in Table 3.4-2. To use the worksheets outside this life-cycle context, the product being evaluated should be matched as closely as possible to products identified in Table 3.4-2, and then the corresponding worksheets can be selected. For example, the technical portion of a proposal might correspond closest to a system and/or system segment specification or to a software requirements specification. Worksheet 0 and/or 1 would be chosen and appropriate questions selected. When the source code is available for an operational product, worksheets 4A and 4B would be used. If the detailed design documentation were available, worksheets 3A and 3B would also be used. Data collection and analysis results can be reported using the Software Quality Evaluation Report (see App. C).

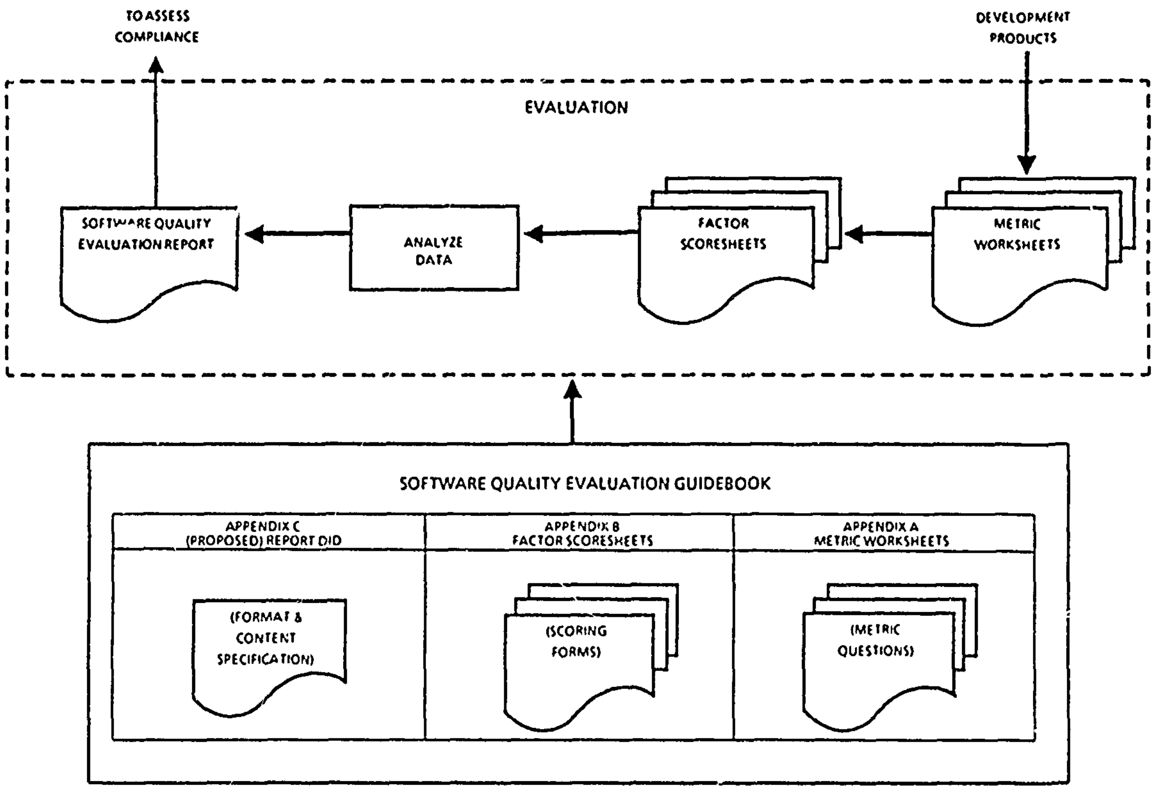


Figure 4.0-3 Metric Data Collection, Analysis, and Reporting

Evaluation Procedural Overview. Achieved quality levels should be evaluated near the end of each development phase for the product sets listed in Table 3.4-2. Achieved quality levels can be reevaluated each time a new version of a document in the product set is released. Reevaluation can be performed by repeating all evaluation procedures or by repeating selected procedures and updating prior results.

The general flow of information for evaluating achieved quality levels is depicted in Figure 4.0-3. Development products are used as source material for answering questions on metric worksheets. Answers on worksheets are used to score metric elements on factor scoresheets, and scores are calculated for the parent metrics, criteria, and factor according to the hierarchical relationship indicated on the scoresheet. Scoring results are compared to requirements and variations analyzed. All results are documented in a Software Quality Evaluation Report and submitted to the SPO.

Appendix A contains metric worksheets. Appendix B contains factor scoresheets. Worksheets and scoresheets are in a general format and require tailoring for each project development phase. Appendix C contains the proposed data item description (DID) for reporting the results of evaluation. Specific content will vary for each project and for each development phase.

Worksheets are applied at different levels as indicated in Table 3.4-1. Worksheet 0 is applied at the system or system segment level. Worksheets 1, 2, 3A, and 4A are applied at the CSCI level. Worksheets 3B and 4B are applied at the unit level. Answers from unit-level worksheets (3B and 4B) are only used for answering questions on corresponding CSCI-level worksheets (3A and 4A) and are not used in conjunction with scoresheets. Only system-level and CSCI-level worksheet answers are used to score metric elements on factor scoresheets as indicated in Figure 4.0-4.

Software quality evaluation is divided into two separate processes (as is shown in Fig. 4.0-1): score quality aspects and interpret quality scores. There are three procedures for scoring quality aspects, as shown in Figure 4.0-5: (1) identify allocation relationships, (2) apply worksheets, and (3) score factors. Sections 4.1, 4.2, and 4.3 describe the details of each procedure.

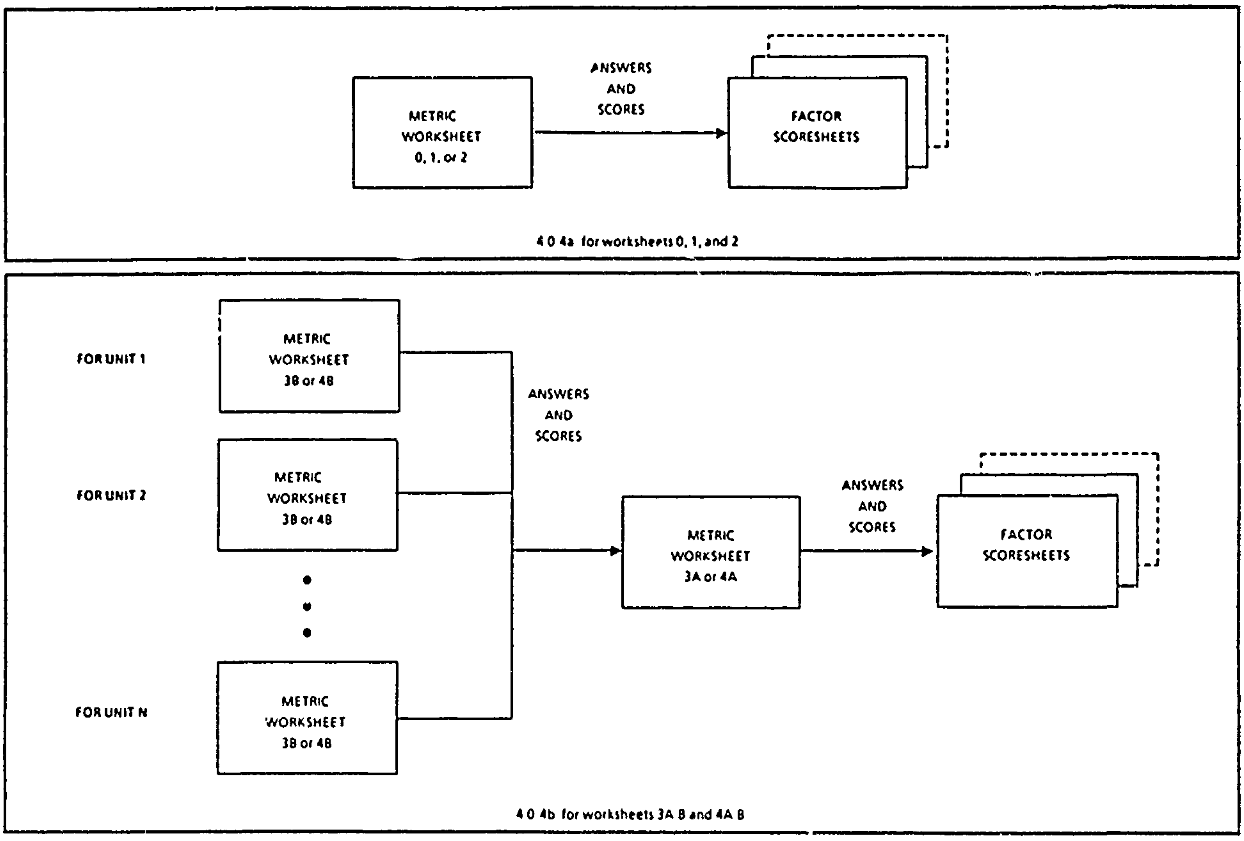


Figure 4.0-4 Metric Worksheet and Factor Scoresheet Correlation

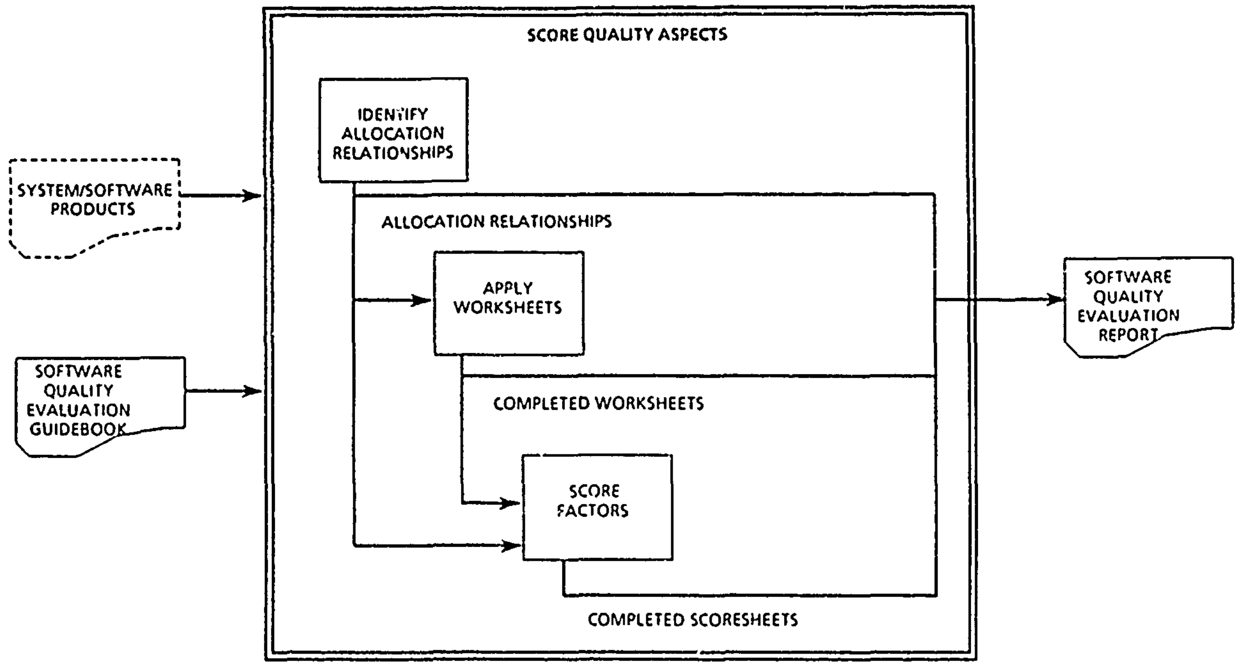


Figure 4.0-5 Procedures for Scoring Quality Aspects

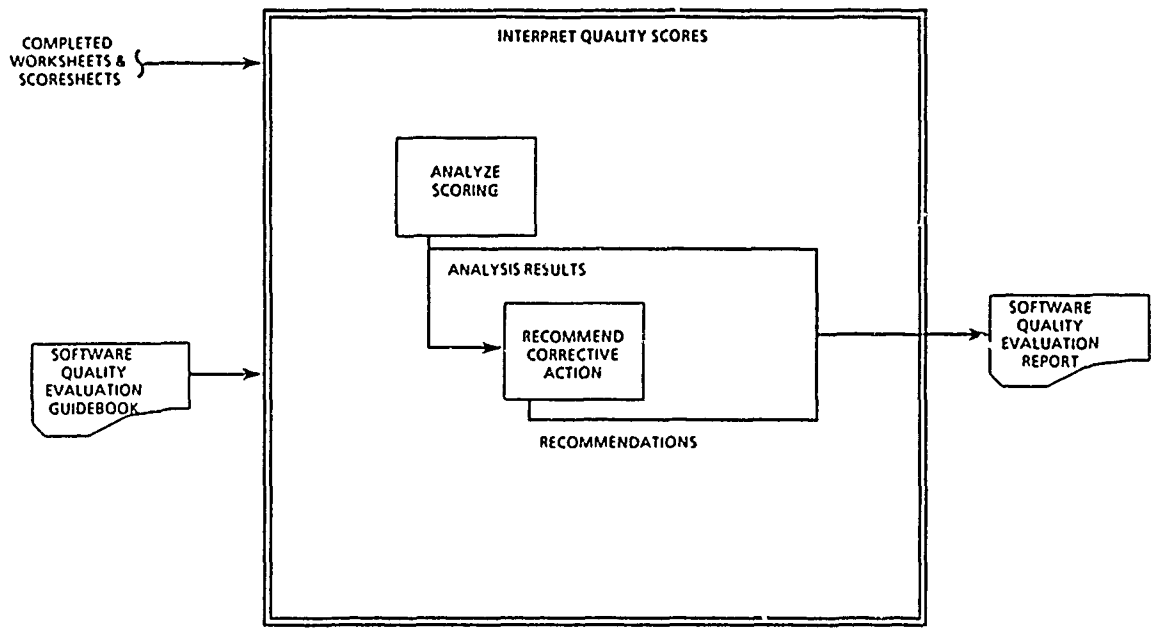


Figure 4.0-6 Procedures for Interpreting Quality Scores

There are two procedures for interpreting quality scores, as shown in Figure 4.0-6: (1) analyze scoring and (2) recommend corrective action. Sections 4.4 and 4.5 describe the details of each procedure.

Procedures assume that software quality requirements have been specified quantitatively in a system-level specification and include factor goals, attribute criteria, criteria weighting, and metrics applicable to government evaluations.

4.1 IDENTIFY ALLOCATION RELATIONSHIPS

This is the first of three procedures for scoring quality aspects. The purpose of this procedure is to derive the relationships between system-level functions and software elements to which quality requirements have been allocated. These relationships define how metric scores for development products should be used for scoring quality factors for each system-level function.

Each relationship should reflect the allocation of quality requirements to software elements, and scoring should show how each applicable software element contributes to an aspect of quality for the system-level function. Allocations of quality requirements should parallel allocations of technical performance and design requirements because the quality requirements are associated with performing specific system-level functions. Each software element supporting a specific system-level function should be allocated quality requirements associated with that function. Many specifications contain allocation and/or traceability matrices. These matrices can aid in deriving specific relationships.

CSCI level. Evaluation formulas should be used to show the relationship between system-level functions and CSCIs to which quality requirements have been allocated. Evaluation formulas are used to relate factor scores for each CSCI to factor scores for the parent function. One evaluation formula is required for each system-level function to show how each applicable CSCI contributes to quality aspects of the system-level function. A table can be used to simplify formula derivation. All system-level functions and CSCIs for which quality requirements have been specified are listed on separate axes. An X is used to indicated each CSCI to which quality requirements have been allocted from a specific function. The evaluation formula for

each system-level function can be derived from the table and would indicated that system-level factor scores are the average of applicable CSCI factor scores.

An average score is used in circumstances where the contribution of each CSCI is approximately the same. If CSCI sizes vary significantly or quality requirements are allocated to only small portions of certain CSCIs, using an average score will bias results, and, therefore, a weighting scheme should be used. A number from zero to one should be used in place of an X in the table and should indicate the percent contribution of each CSCI by relative amount of applicable code. Sizing estimates should be used during design phases and actuals during subsequent phases.

Unit level. During the detailed design and the coding and unit testing phases, metric scoring is also performed at the unit level for each CSCI. It is not necessary to derive relationship formulas because answers for unit-level worksheets are used in filling out CSCI-level worksheets. Applicable scoring at the CSCI level reflects average scoring at the unit level. (Units are assumed to be approximately equal in size.) However it is necessary to identify units to which quality requirements have been assigned to enable application of unit-level worksheets to appropriate units. A list of applicable units should be made for each CSCI. If a CSCI has been allocated quality requirements from more than one system-level function, a separate list of applicable units should be made for each applicable function.

Results of this procedure should be submitted for inclusion in the Software Quality Evaluation Report.

4.2 APPLY WORKSHEETS

This is the second of three procedures for scoring quality aspects (see Fig. 4.0-5). The purpose of this procedure is to collect metric data using worksheets. There are seven metric worksheets, organized by development phase, contained in Appendix A. Worksheets are applied near the end of a development phase to the products of that phase.

This procedure consists of three steps:

a. Prepare worksheets (step 1).

b. Gather source material (step 2).

c. Answer worksheet questions (step 3).

4.2.1 Prepare Worksheets (Step 1)

Metric worksheet preparation is required prior to collecting data. The appropriate worksheets should be selected from Appendix A. Table 3.4-1 shows the development phases during which worksheets are applied. Worksheet 0 is used during system/software requirements analysis, worksheet 1 during software requirements analysis, and worksheet 2 during preliminary design. Worksheets 3A and 3B are used during detailed design; 3B is applied at the unit level and 3A at the CSCI level. Worksheets 4A and 4B are used during code and unit testing; 4B is applied at the unit level and 4A at the CSCI level. During the test and integration phases, selected metric questions are reapplied as indicated in the quality attribute correlation table in Appendix A. The primary purpose is to determine that answers to selected questions have not changed as a result of implementation or testing. Answers to these questions can be monitored individually and compared to answers from prior phases or can be used with applicable answers from prior phases to calculate new factor scores.

Each worksheet contains questions regarding all quality attributes applicable to the phase during which the worksheet is applied. Worksheets should be tailored for the specific quality attributes of interest. It would be unusual to have quality requirements for all 13 factors specified for any one system-level function. A subset of quality aspects is normally specified for each function and therefore for the software supporting that function. Worksheets should be tailored to reflect only quality attributes of interest by deleting unneeded questions. The system-level specification should contain a list of applicable criteria and should identify metrics that will be used by the government for evaluating achieved quality levels. This information can be used as a basis for deleting unneeded questions on worksheets. The quality attribute correlation table in Appendix A identifies the criteria, metrics, and metric elements applicable to each phase and indicates which metric elements have corresponding questions for each worksheet.

System-level. One worksheet is required for each applicable system-level function. Each worksheet should be tailored for the specific quality attributes of interest for that function.

CSCI level. One worksheet is required for each CSCI identified in Section 4.1. Each worksheet should be tailored for the specific quality attributes of interest for that CSCI. If a CSCI has been allocated quality requirements from more than one system-level function, attributes of interest should correspond to quality requirements for all applicable functions. The table developed in Section 4.1 identifies each function from which quality requirements have been allocated to a CSCI.

Unit level. One worksheet is required for each applicable unit identified in Section 4.1. Each worksheet should be tailored for the specific quality attributes of interest for that unit. If the parent CSCI has been allocated quality requirements from only one system-level function, the quality attributes of interest are the same for the unit as for the parent CSCI. If the parent CSCI has been allocated quality requirements from more than one system-level function, the unit may have been assigned quality requirements from any one or more functions. Lists of applicable units, by CSCI, were developed for each function in Section 4.1. Attributes of interest for each unit should correspond to quality requirements for all applicable functions. Each unit should have only one worksheet.

4.2.2 Gather Source Material (Step 2)

Source material should be gathered for answering questions on metric worksheets. Table 3.4-2 lists the minimum product set by software development phase that is required for answering worksheet questions. Each product is identified by title and by DID number. Information from the entire set of products for a particular phase is needed to answer questions on the worksheets applicable to that phase. It is not necessary to have the specific product set listed, only to have equivalent information available for answering questions. For example, the QA plan and software standards may be included as part of the software development plan for smaller projects.

4.2.3 Answer Worksheet Questions (Step 3)

Metric worksheets are divided into two sections—A and B. Section A is for general information describing the specifics of an application of the worksheet and should remain with the rest of the worksheet for identification. Section B contains metric questions. Questions are organized alphabetically and numerically by metric element acronym, as indicated in the quality attribute correlation table in Appendix A. Terminology used in the worksheets generally is consistent with terminology in proposed DOD-STD-SDS (e.g., CSCI, unit). The term "software" is used in a general sense throughout the worksheets and refers both to the end product (code, data) and to the product in its most recent stage of evolutionary development; i.e., to the software as represented by the product set for a development phase.

Worksheet questions should be answered by personnel familiar with the project software, software documentation scheme, QM technology, and general concepts and terminology used in DOD-STD-SDS. Questions should be answered objectively. Examples and explanatory information are included with many questions, and a glossary is provided at the end of each worksheet. Some questions may require subjective judgement. To minimize variations in answers, review parent criterion and factor definitions for each metric (see Sec. 3.0), and review all questions applicable to one metric prior to answering the first question.

Identifying source material for answering any one question should be simple for personnel familiar with the project documentation scheme. For example, for questions regarding the system or software itself, refer to specifications or design documents; and for questions regarding standardization, refer to documents describing software standards. If source material is not available for answering questions, questions should be answered in the negative as this is a deficiency. Answers and scoring should simply reflect the degree of system and software characteristics.

Many worksheet questions use the terms "all" or "free from". These terms are intended as totally inclusive and totally exclusive, respectively. Any one instance of variation requires that the question be answered in the negative to bring attention to a variation. Variations are potential problems, and part of the intent of QM technology is to detect problems early in the development cycle when corrections are less costly.

Some worksheet questions may be redundant among worksheets for some projects. For example, if two CSCIs have been allocated requirements from the same parent specification, some requirements may be the same (e.g., interface protocol, data format); and if one development contractor or subcontractor is developing all CSCIs in the same language, the same development standards may be used. Regardless of circumstances, all applicable questions on each worksheet should be answered because each worksheet is used separately when scores are calculated.

Results of applying worksheets should be reproducible; that is, if the same worksheet is applied by a second person, answers should be the same. In cases where judgement is exercised in answering worksheet questions, rationale should be noted. In cases where violations occur, the software element in violation should be noted. These and other applicable comments should be compiled with the completed worksheet and submitted by memo in a format suitable for inclusion in the Software Quality Evaluation Report (see App. C).

Normally, a worksheet will be applied at least twice—once to the draft version and once to the final version. Data collection personnel may find it efficient to also record additional information such as paragraphs referenced for answering questions.

4.3 SCORE FACTORS

This is the third of three procedures for scoring quality aspects (see Fig. 4.0-5). The purpose of this procedure is to calculate scores for each software quality factor using factor scoresheets and information from completed metric worksheets. There are 13 factor scoresheets—one for each software quality factor—contained in Appendix B. Worksheet information is transferred to scoresheets at the metric element level. Scores are then calculated for the parent metrics, criteria, and factor according to the hierarchical relationship indicated on the scoresheet.

This procedure consists of two steps:

a. Prepare scoresheets (step 1).
b. Calculate factor scores (step 2).

4.3.1 Prepare Scoresheets (Step 1)

Factor scoresheet preparation is required prior to calculating scores. The appropriate factor scoresheets should be selected from Appendix B. One scoresheet should be selected for each quality factor of interest. At the system level, one scoresheet is required for each factor of each system-level function. At the CSCI level, one scoresheet is required for each factor applicable to each CSCI. CSCIs and parent functions are identified in the table developed in Section 4.1. Quality factors of interest should correspond to quality factor requirements allocated from system-level functions. If a CSCI has been allocated quality requirements from more than one function, factors of interest should correspond to quality requirements for all applicable functions.

Factor scoresheets should be tailored to reflect only quality attributes (criteria, metrics, and metric elements) of interest by deleting unnecessary attributes. The system-level specification should contain a list of applicable criteria and should identify metrics that will be used by the government for evaluating achieved quality levels. This information can be used as a basis for deleting unnecessary attributes on scoresheets. This process is similar to that performed in tailoring metric worksheets in that source information is the same. The primary differences are:

a. For each CSCI there is only one worksheet but up to 13 scoresheets.
b. Quality attributes appear only once on a given worksheet; attributes may appear on more than one scoresheet because of criterion/factor relationships (see Tbl. 3.2-1).

The same factor scoresheet templates are used for each development phase. Each scoresheet needs further tailoring to make it compatible with the current development phase. The quality attribute correlation table in Appendix A identifies the criteria, metrics, and metric elements applicable to each phase. Nonapplicable attributes should be deleted from each scoresheet.

The end result of scoresheet tailoring should be scoresheets that are compatible with worksheets for the current development phase. For each scoresheet metric element, there should be an answer to the corresponding metric element question on the corresponding worksheet. And for each worksheet answer, there should be a corresponding scoresheet metric element on each applicable scoresheet.

Each criterion is assigned a weighting value when calculating factor scores. Separate weighting formulas are used for each factor. If a criterion is an attribute of more than one factor, a different weighting value may be assigned for each parent factor. The system-level specification should list factor weighting formulas required for scoring. Weighting values should be entered on each factor scoresheet for easy referencing when performing calculations.

4.3.2 Calculate Factor Scores (Step 2)

Scoresheets contain blanks for entering the factor score and scores for each attribute: criteria, metrics, and metric elements. Scoring starts at the metric element level. Answers to each worksheet question are translated into a metric element score, and the score is entered in the corresponding metric element blank on the scoresheet. "Yes" answers are scored 1. "No" answers are scored zero. Numeric answers are transferred directly scoresheets (range from zero to one).

After all metric element scores have been entered, metric scores are calculated. Metric scores are the average of all applicable metric element scores. Criteria scores are calculated next and are the average of all applicable metric scores. One factor score is calculated for each scoresheet using the factor weighting formula. Criteria weighting values should sum to 1.0.

Completed scoresheets and pertinent comments should be submitted for inclusion in the Software Quality Evaluation Report with the worksheet used as source material.

4.4 ANALYZE SCORING

This is the first of two procedures for interpreting quality scores (see Fig. 4.0-6). In this procedure, scoring results are used to determine variations from specified requirements and causes of variations. Scoresheets and worksheets from previous procedures are used as source material. Results are documented in the Software Quality Evaluation Report.

This procedure consists of four steps:

a. Calculate functional scores (step 1).
b. Compute scoring trends (step 2).
c. Compare scores with requirements (step 3).
d. Analyze variations (step 4).

4.4.1 Calculate Functional Scores (Step 1)

Step 1 of this procedure is to calculate scores for each quality factor for which requirements have been specified at the system level. A separate factor score is calculated for each factor of each applicable system-level function. Scoresheet results are used as source information.

If the current phase is system/software requirements analysis, factor scores on scoresheets correspond to factor scores for each system-level function, and no further action is required.

If the current phase is other than system/software requirements analysis, factor scores on scoresheets are for a CSCI and may not correspond to factor scores for functions. The evaluation formulas developed in Section 4.1 show the relationship between each system-level function and each applicable CSCI. Use the evaluation formula and scoresheet factor scores and calculate functional factor scores for each system-level function. The same formula is used for all factors of one function.

4.4.2 Compute Scoring Trends (Step 2)

Step 2 of this procedure is to compute factor scoring trends over development phases. Scoring results from previous evaluations and from this evaluation should be used as data points to show the scoring trend for each factor. At a minimum, functional factor scores should be plotted. Plots of other scoring trends may prove useful in analyzing scoring (e.g., CSCI factor scores).

4.4.3 Compare Scores with Requirements (Step 3)

Step 3 of this procedure is to compare scores computed in step 1 to specified requirements. The purpose of this step is to determine if scores satisfy specified requirements. Scores range from zero to one. Requirements are normally specified as a quantitative range (e.g., from 0.80 to 0.89). Graphs can be used to pictorially compare goals with scores. At a minimum, comparisons should group all factors for a single system-level function. Other groupings and comparisons may prove useful in analyzing scoring (e.g., grouping the same factor for all applicable system-level functions and comparing goals and scores at the CSCI level).

4.4.4 Analyze Variations (Step 4)

Step 4 of this procedure is to identify each factor for which scoring goals are not satisfied and to explore the cause. This step deals only with specific causes for scoring deficiencies. Remedies for correcting scoring deficiencies are explored in the next section.

All scoring deficiencies for each system-level function should be identified. The process of analysis to determine causes should start at the system level and proceed to lower levels (CSCI and unit) incrementally. Analysis at each level should provide clues as to which areas to explore at the next-lower level to determine causes of scoring deficiencies. Analysis should include investigating scoring patterns because these are especially revealing.

For example, analysis of scoring at the system level could reveal that a factor score is consistently low for all functions. If so, analysis at the CSCI level would likely show the same pattern for the same factor. Investigation of attribute scoring should show which attributes are low-scoring, and a review of low scoring metrics should show the nature of the basic problem.

Another possibility is that analysis of system-level scores reveals that scoring is consistently low for one function. Further investigation should focus on scoring for CSCIs supporting that function. CSCI-level investigation should show which attributes are low scoring. If only specific attributes are low scoring, a review of low scoring

metrics should show the nature of the problem. If attributes are low scoring in general, the cause could be a low quality product or that methods used by data collection personnel are resulting in low scores. Appropriate products and completed worksheets should be reviewed.

If worksheets 3A or 4A were used for CSCI-level scoring, low scoring can be traced through worksheet answers to the unit level (worksheets 3B and 4B) to determine the cause of scoring patterns.

Results of this analysis should be submitted for inclusion in the Software Quality Evaluation Report. Causes should be listed separately for each scoring deficiency.

The following are example causes:

a. The software design does not provide this characteristic.
b. This characteristic is only required for portions of the design.
c. Development standards do not refer to this characteristic.
d. Development practices vary among design teams; no standard has been established in this area (or, the standard is interpreted differently).
e. Information for this characteristic was not documented.
f. This characteristic is provided by most but not all software elements. The worksheet question was worded "all" and therefore answered in the negative. Exceptions are noted here and explained on the corresponding worksheets.
g. This characteristic was not provided for this software element because to do so would be in conflict with derived (or other) requirements.
h. Scores for this characteristic varied considerably because personnel answering worksheet questions could not agree on its interpretation for this project.
i. This characteristic is not available for this project because of language/compiler limitations.
j. This characteristic was most often scored negative but does not seem to apply to this project (e.g., no technical requirement).
l. This area of the design/code consistently scored low for the following characteristics . . . but there is no violation of technical requirements.

4.5 RECOMMEND CORRECTIVE ACTION

This is the second of two procedures for interpreting quality scores (see Fig. 4.0-6). The purpose of this procedure is to summarize problems and provide recommendations for correcting scoring deficiencies defined in the previous procedure (see Sec. 4.4). Results are documented in the Software Quality Evaluation Report.

This procedure consists of two steps:

a. Summarize problems (step 1).
b. Provide recommendations (step 2).

4.5.1 Summarize Problems (Step 1)

Step 1 of this procedure is to provide a summary of problems. Analyses results from the previous procedure (see Sec. 4.4) are summarized to the level of problem statements and are used in the next step as bases for recommending corrective action.

The lists of deficiencies and causes developed in the previous procedure should be reviewed. Causes are stated in specific terms. Causes should be categorized so that a more general statement of the problem can be formulated. These summary statements should clearly define the problem at a level where action could be taken. Several iterations on categorization may be required before satisfactory problem statements are developed.

The following are example problem statements:

a. There are no technical requirements to provide certain features or characteristics measured by the metrics.
b. Development standards are not strictly enforced or are vague.
c. Design or implementation choices do not provide certain characteristics measured by the metrics.
d. Development standards are not as comprehensive as metric questions.
e. There are conflicts between certain technical performance or design requirements and the quality requirements.
f. The documentation scheme does not provide information required to answer certain metric questions.

g. Requirements, design, or code are deficient in certain areas.
h. Certain metric scores indicate that requirements have not been allocated appropriately.
i. Certain scores are low because of necessary design or implementation exceptions.

4.5.2 Provide Recommendations (Step 2)

The purpose of this step is to provide recommendations for correcting scoring deficiencies. Summary problem statements from step 1 are used as source material. Results are documented in the Software Quality Evaluation Report.

Specific recommendations for corrective action should be provided whenever possible and should include rationale. Otherwise, provide summaries of action alternatives with potential results and consequences. Recommendations and action alternatives should focus on technical solutions to problem statements and should not consider administrative issues (e.g., action responsibility, schedule, and budget).

A broad range of actions is possible. Action recommendations and alternatives should reflect realistic solutions for all problem statements and should consider system goals and requirements and design and implementation limitations and constraints.

The following are example actions:
a. Change certain quality requirements (e.g., add, modify, or delete quality attributes or change goal levels).
b. Change certain quality attribute measurements or scoring (e.g., monitor selected attribute scores but do not use in scoring).
c. Change certain technical performance or design requirements.
d. Change allocation of certain requirements (technical or quality).
e. Change certain aspects of the documentation scheme.
f. Change certain aspects of standards or enforce standards.
g. Correct certain requirement, design, or implementation deficiencies.

4.6 AUTOMATION

Portions of the evaluation process can be automated for the sake of efficiency. Worksheet information and answers could be stored in a data base, and reasonableness checks could be performed on answer entries. Factor scoring, comparisons with requirements, and trend analyses could be performed automatically. Scoring pattern recognition could be computer-aided. And report generation could be automated. It is also possible to automatically scan source information (e.g., code and PDL) to obtain answers for some metric questions.

APPENDIX A
METRIC WORKSHEETS

Appendix A contains metric worksheets used to collect metric data during development phases. Seven different worksheets are applied to development products in different phases and at different levels of abstraction.

a. Metric Worksheet 0, system level, system/software requirements analysis.
b. Metric Worksheet 1, CSCI level, software requirements analysis.
c. Metric Worksheet 2, CSCI level, preliminary design.
d. Metric Worksheet 3A, CSCI level, detailed design.
e. Metric Worksheet 3B, unit level, detailed design.
f. Metric Worksheet 4A, CSCI level, code and unit testing.
g. Metric Worksheet 4B, unit level, code and unit testing.

Selected metric questions are reapplied during subsequent phases as indicated in the quality attribute correlation table on the next few pages.

Worksheets are divided into two Sections—A and B. Section A is for general information identifying a specific application of the worksheet. Section B contains metric elements in question format. Th applicable metric element is identified by acronym at the beginning of each worksheet question. Each worksheet contains questions for all metric elements applicable to the phase during which it is applied. Questions on each worksheet are organized alphabetically and numerically by metric element acronym. The quality attribute correlation table on the next several pages identifies each metric element and parent metric and criterion and summarizes the metric element content of each worksheet.

Terminology used in the worksheets generally is consistent with proposed DOD-STD-SDS (e.g., CSCI, unit). The term "software" is used in a broad sense and refers both to the end product (code, data) and to the product in its most recent stage of evolutionary development. For example, on Metric Worksheet 1, software refers to descriptions of the software in the software requirements specification and other documents that are products of the requirements analysis phase. A glossary is provided at the end of each Section B.

METRIC WORKSHEET/QUALITY ATTRIBUTE CORRELATION

Quality factor attributes \ Life-cycle phase/activity			Demonstration and Validation		Full Scale Development					
Criteria	Metrics	Metric elements	System/ Software Req'mts Analysis	Software Req'mts Analysis	Preliminary Design	Detailed Design	Coding and Unit Testing	CSC Integration and Testing	CSCI-level Testing	System Integration and Testing
Accuracy	Accuracy checklist	AC.1(1)	0							
		AC.1(2)	0							
		AC.1(3)	0	1						
		AC.1(4)	0	1						
		AC.1(5)	0	1						
		AC.1(6)	0	1						
		AC.1(7)			2			2	2	2
		AC.1(8)					4A	4A		
Anomaly Mangement	Error tolerance/ control	AM.1(1)	0	1					1	0
		AM.1(2)	0	1					1	0
		AM.1(3)	0	1		3A,B	4A,B			
		AM.1(4)	0	1					1	0
	Improper input data	AM.2(1)	0	1						
		AM.2(2)				3A	4A			
		AM.2(3)				3A	4A			
		AM.2(4)				3A	4A			
		AM.2(5)				3A	4A			
		AM.2(6)				3A	4A	4A		
		AM.2(7)				3A,B	4A,B			
	Computational failures	AM.3(1)	0	1	2					
		AM.3(2)		1		3A	4A			
		AM.3(3)		1		3A	4A			
		AM.3(4)		1		3A	4A	4A		
	Hardware faults	AM.4(1)	0	1	2			2		
	Device errors	AM.5(1)	0	1	2			2		
	Communication errors	AM.6(1)	0	1	2			2	2	
		AM.6(2)			2			2	2	
		AM.6(3)			2			2	2	
		AM.6(4)			2			2	2	
	Node/ communication failures	AM.7(1)	0	1	2			2		
		AM.7(2)	0	1	2			2		
		AM.7(3)	0	1	2			2		
Application Independence	Database management implementation independence	AP.1(1)	0	1	2	3A,B	4A,B			
	Data structure	AP.2(1)				3A,B	4A,B			
		AP.2(2)	0	1		3A,B	4A,B			
		AP.2(3)				3A,B	4A,B			
		AP.2(4)	0	1		3A,B	4A,B			
	Architecture standardization	AP.3(1)	0	1		3A,B	4A,B			
		AP.3(2)					4A,B			
	Microcode independence	AP.4(1)	0	1		3A,B	4A,B			
	Functional independence	AP.5(1)	0	1	2					
		AP.5(2)			2			2		
		AP.5(3)			2			2		
Augmentability	Data storage expansion	AT.1(1)				3A,B	4A,B			
		AT.1(2)	0	1	2	3A	4A	4A	4A	
		AT.1(3)	0	1	2	3A	4A	4A	4A	
	Computation extensibility	AT.2(1)				3A,B	4A,B			
		AT.2(2)				3A,B	4A,B			
		AT.2(3)	0	1	2	3A	4A	4A	4A	
	Channel extensibility	AT.3(1)	0	1	2	3A	4A	4A	4A	
		AT.3(2)	0	1	2	3A	4A	4A	4A	
	Design extensibility	AT.4(1)	0	1	2			2	2	
		AT.4(2)	0	1						
		AT.4(3)	0	1						
Autonomy	Interface complexity	AU.1(1)	0	1	2					
		AU.1(2)				3A	4A			
		AU.1(3)				3A	4A			
		AU.1(4)			2	3A	4A	4A	4A	
	Self-sufficiency	AU.2(1)	0	1					1	1
		AU.2(2)	0	1	2			2	2	

METRIC WORKSHEET/QUALITY ATTRIBUTE CORRELATION (CONTINUED)

Quality factor attributes / Life-cycle phase/activity			Demonstration and Validation	Full Scale Development						
Criteria	Metrics	Metric elements	System/ Software Req'mts Analysis	Software Req'mts Analysis	Preliminary Design	Detailed Design	Coding and Unit Testing	CSC Integration and Testing	CSCI-level Testing	System Integration and Testing
Commonality	Communications commonality	CL.1(1)	0	1						
		CL.1(2)	0	1	2					
		CL.1(3)	0	1	2			2		
		CL.1(4)	0	1	2			2		
		CL.1(5)	0	1	2			2		
		CL.1(6)	0	1	2			2		
		CL.1(7)	0	1	2	3A	4A	2	1	0
		CL.1(8)	0	1	2	3A	4A	2	1	0
		CL.1(9)	0	1					1	0
		CL.1(10)	0	1					1	0
		CL.1(11)	0	1	2			2	1	0
		CL.1(12)	0	1					1	0
		CL.1(13)	0	1	2				1	0
		CL.1(14)	0	1	2				1	0
	Data commonality	CL.2(1)	0	1	2	3A	4A	2		
		CL.2(2)	0	1	2	3A	4A	2	1	0
		CL.2(3)	0	1						
		CL.2(4)	0	1	2				1	0
		CL.2(5)	0	1						
		CL.2(6)	0	1	2				1	0
		CL.2(7)	0	1					1	0
		CL.2(8)	0	1					1	0
	Common vocabulary	CL.3(1)	0	1						
Completeness	Completeness checklist	CP.1(1)	0	1	2	3A,B				
		CP.1(2)	0	1	2	3A,B	4A,B			
		CP.1(3)	0	1	2	3A	4A			
		CP.1(4)			2	3A,B	4A,B			
		CP.1(5)	0	1						
		CP.1(6)	0	1	2					
		CP.1(7)	0	1						
		CP.1(8)	0	1						
		CP.1(9)			2	3A,B	4A,B			
		CP.1(10)				3A,B	4A,B			
		CP.1(11)	0	1	2	3A	4A	2	1	0
Consistency	Procedure consistency	CS.1(1)	0	1	2	3A,B				
		CS.1(2)	0	1		3A,B	4A,B			
		CS.1(3)	0	1		3A,B	4A,B			
		CS.1(4)	0	1		3A,B	4A,B			
		CS.1(5)	0	1	2	3A,B	4A,B			
	Data consistency	CS.2(1)	0	1	2	3A,B	4A,B			
		CS.2(2)	0	1	2	3A,B	4A,B			
		CS.2(3)	0	1	2	3A,B	4A,B			
		CS.2(4)	0	1	2					
		CS.2(5)	0	1	2					
		CS.2(6)	0	1	2	3A,B	4A,B			
Distributedness	Design structure	DI.1(1)	0	1	2	3A				
		DI.1(2)	0	1	2					
		DI.1(3)	0	1						
		DI.1(4)	0	1	2			2	2	2
		DI.1(5)	0		2			2	2	2
		DI.1(6)	0	1	2			2	2	2
		DI.1(7)	0	1	2			2	2	2
		DI.1(8)	0	1						
		DI.1(9)	0	1	2			2	2	2
Document Accessibility	Access to documentation	DO.1(1)	0	1	2	3A	4A	2	1	0
	Well-structured documentation	DO.2(1)	0	1	2	3A	4A	2	1	0
		DO.2(2)	0	1	2	3A				
		DO.2(3)	0	1	2	3A	4A	2	1	0
		DO.2(4)	0	1	2	3A	4A	4A	4A	4A
		DO.2(5)	0	1	2					
		DO.2(6)	0	1						
		DO.2(7)			2	3A				
		DO.2(8)					4A	4A	4A	4A

METRIC WORKSHEET/QUALITY ATTRIBUTE CORRELATION (CONTINUED)

Quality factor attributes \ Life-cycle phase/activity			Demonstration and Validation		Full Scale Development					
Criteria	Metrics	Metric elements	System/ Software Req'mts Analysis	Software Req'mts Analysis	Preliminary Design	Detailed Design	Coding and Unit Testing	CSC Integration and Testing	CSCI-level Testing	System Integration and Testing
Effectiveness-Communication	Communication effectiveness measure	EC.1(1)	0	1						
Effectiveness-Processing	Processing effectiveness measure	EP.1(1)	0	1						
		EP.1(2)				3A,B	4A,B			
		EP.1(3)	0	1		3A	4A,B			
		EP.1(4)				3A,B	4A,B			
		EP.1(5)	0	1	2	3A	4A	4A	4A	4A
		EP.1(6)				3A,B	4A,B			
	Data usage effectiveness measure	EP.2(1)	0	1						
		EP.2(2)	0	1	2	3A				
		EP.2(3)	0	1	2					
		EP.2(4)				3A,B	4A,B			
		EP.2(5)				3A,B	4A,B			
		EP.2(6)			2	3A				
		EP.2(7)				3A,B	4A,B			
Effectiveness-Storage	Storage effectiveness measure	ES.1(1)	0	1						
		ES.1(2)	0	1	2					
		ES.1(3)				3A	4A			
		ES.1(4)				3A	4A			
		ES.1(5)	0	1	2					
		ES.1(6)				3A,B	4A,B			
		ES.1(7)	0	1			4A			
		ES.1(8)	0	1	2	3A	4A			
Functional Overlap	Functional overlap checklist	FO.1(1)	0	1						
		FO.1(2)	0	1						
		FO.1(3)	0	1						
		FO.1(4)	0	1						
Functional Scope	Function specificity	FS.1(1)				3A,B	4A,B			
		FS.1(2)					4A,B			
	Function commonality	FS.2(1)	0	1						
		FS.2(2)	0	1	2					
		FS.2(3)	0	1	2					
		FS.2(4)	0	1	2					
		FS.2(5)	0	1	2					
		FS.2(6)	0	1						
	Function selective usability	FS.3(1)	0	1	2				2	
		FS.3(2)	0	1	2				2	
		FS.3(3)	0	1						
Generality	Unit referencing	GE.1(1)				3A	4A			
	Unit implementation	GE.2(1)				3A	4A			
		GE.2(2)				3A,B	4A,B			
		GE.2(3)				3A,B	4A,B			
		GE.2(4)				3A,B	4A,B			
Independence	Software independence from system	ID.1(1)				3A,B	4A,B			
		ID.1(2)	0	1						
		ID.1(3)	0	1	2	3A,B	4A,B			
	Machine independence	ID.2(1)	0	1	2					
		ID.2(2)				3A	4A			
		ID.2(3)				3A	4A			
		ID.2(4)				3A	4A			
Modularity	Modular implementation	MO.1(1)	0	1						
		MO.1(2)	0	1	2	3A	4A			
		MO.1(3)				3A,B	4A,B			
		MO.1(4)				3A,B	4A,B			
		MO.1(5)				3A,B	4A,B			
		MO.1(6)				3A,B	4A,B			
		MO.1(7)				3A,B	4A,B			
		MO.1(8)				3A,B	4A,B			
		MO.1(9)			2	3A,B	4A,B			
	Modular design	MO.2(1)	0	1						
		MO.2(2)		1	2	3A	4A			
		MO.2(3)		1	2	3A	4A			
		MO.2(4)	0	1						
		MO.2(5)		1	2	3A,B	4A,B			

METRIC WORKSHEET/QUALITY ATTRIBUTE CORRELATION (CONTINUED)

Quality factor attributes \ Life-cycle phase/activity			Demonstration and Validation		Full Scale Development					
Criteria	Metrics	Metric elements	System/ Software Req'mts Analysis	Software Req'mts Analysis	Preliminary Design	Detailed Design	Coding and Unit Testing	CSC Integration and Testing	CSCI-level Testing	System Integration and Testing
Operability	Operability checklist	OP.1(1)	0	1	2				2	2
		OP.1(2)	0	1	2	3A	4A		4A	4A
		OP.1(3)	0	1	2	3A	4A		2	2
		OP.1(4)	0	1	2				2	2
		OP.1(5)	0	1						0
		OP.1(6)	0	1						
		OP.1(7)			2				2	2
		OP.1(8)	0	1						
		OP.1(9)			2				2	2
		OP.1(10)	0	1	2	3A	4A		2	2
		OP.1(11)	0	1	2				2	2
		OP.1(12)	0	1	2				2	2
		OP 1(13)	0	1	2				2	2
		OP.1(14)	0	1	2				2	2
		OP.1(15)	0	1	2				2	2
		OP.1(16)	0	1	2				2	2
	User input communicativeness	OP.2(1)	0	1					1	0
		OP.2(2)		1					1	1
		OP 2(3)		1					1	1
		OP.2(4)	0	1	2				2	2
		OP.2(5)	0	1	2				2	
		OP.2(6)	0	1	2				2	2
	User output communicativeness	OP.3(1)	0	1	2				2	2
		OP 3(2)	0	1	2				2	2
		OP.3(3)	0	1	2				2	2
		OP.3(4)		1					1	1
		OP.3(5)		1					1	1
		OP.3(6)	0	1	2				2	2
		OP.3(7)	0	1	2				2	2
		OP.3(8)	0	1	2				2	2
Reconfigurability	Restructure checklist	RE.1(1)	0	1	2				2	2
		RE.1(2)	0	1	2				2	2
		RE.1(3)	0	1	2				2	2
		RE.1(4)	0	1	2				2	2
Self-Descriptiveness	Quantity of comments	SD.1(1)					4A,B			
	Effectiveness of comments	SD.2(1)	0	1			4A,B			
		SD.2(2)	0	1			4A,B			
		SD.2(3)					4A,B			
		SD.2(4)					4A,B			
		SD.2(5)					4A,B			
		SD.2(6)					4A,B			
		SD.2(7)					4A,B			
		SD.2(8)					4A,B			
	Descriptiveness of language	SD 3(1)					4A,B			
		SD.3(2)					4A,B			
		SD.3(3)					4A,B			
		SD.3(4)					4A,B			
		SD.3(5)	0	1		3A,B	4A,B			
		SD.3(6)					4A,B			
Simplicity	Design structure	SI.1(1)	0	1	2	3A				
		SI.1(2)				3A,B	4A,B			
		SI.1(3)				3A,B	4A,B			
		SI.1(4)				3A,B	4A,B			
		SI.1(5)				3A,B	4A,B			
		SI.1(6)			2	3A	4A			
		SI.1(7)			2	3A	4A			
		SI.1(8)	0	1						
		SI.1(9)	0	1	2					
		SI.1(10)			2	3A	4A			
	Structured language or preprocessor	SI.2(1)	0	1	2		4A			
	Data and control flow complexity	SI.3(1)				3A,B	4A,B			
	Coding simplicity	SI.4(1)				3A,B	4A,B			
		SI.4(2)				3A,B	4A,B			
		SI.4(3)				3A,B	4A,B			
		SI.4(4)				3A,B	4A,B			
		SI.4(5)				3A,B	4A,B			
		SI.4(6)					4A,B			
		SI.4(7)					4A,B			
		SI.4(8)					4A,B			
		SI.4(9)					4A,B			
		SI 4(10)					4A,B			
		SI 4(11)					4A,B			
		SI.4(12)					4A,B			
		SI.4(13)	0	1	2		4A,B			
		SI.4(14)				3A	4A			
	Specificity	SI.5(1)				3A,B	4A,B			
		SI.5(2)				3A,B	4A,B			
		SI.5(3)				3A,B	4A,B			
	Halstead's level of difficulty measure	SI 6(1)				3A,B	4A,B			

METRIC WORKSHEET/QUALITY ATTRIBUTE CORRELATION (CONTINUED)

Quality factor attributes \ Life-cycle phase/activity			Demonstration and Validation		Full Scale Development					
Criteria	Metrics	Metric elements	System/ Software Req'mts Analysis	Software Req'mts Analysis	Preliminary Design	Detailed Design	Coding and Unit Testing	CSC Integration and Testing	CSCI-level Testing	System Integration and Testing
System Accessibility	Access control	SS.1(1)	0	1	2				2	2
		SS.1(2)	0	1	2				2	2
		SS.1(3)	0	1	2				2	2
		SS.1(4)	0	1	2				2	2
	Access audit	SS.2(1)	0	1	2				2	2
		SS.2(2)	0	1	2				2	2
System Clarity	Interface complexity	ST.1(1)				3A,B	4A,B			
		ST.1(2)				3A,B	4A,B			
		ST.1(3)				3A,B	4A,B			
		ST.1(4)				3A,B	4A,B			
		ST.1(5)				3A,B	4A,B			
		ST.1(6)				3A	4A			
	Program flow complexity	ST.2(1)				3A,B	4A,B			
		ST.2(2)				3A,B	4A,B			
		ST.2(3)				3A,B	4A,B			
		ST.2(4)				3A,B	4A,B			
		ST.2(5)					4A,B			
	Application functional complexity	ST.3(1)	0	1						
		ST.3(2)	0	1	2	3A				
		ST.3(3)				3A,B	4A,B			
		ST.3(4)				3A	4A			
		ST.3(5)			2					
		ST.3(6)			2					
	Communication complexity	ST.4(1)				3A,B	4A,B			
		ST.4(2)				3A,B	4A,B			
		ST.4(3)				3A	4A			
		ST.4(4)				3A,B	4A,B			
		ST.4(5)				3A,B	4A,B			
	Structure clarity	ST.5(1)				3A,B	4A,B			
		ST.5(2)				3A,B	4A,B			
		ST.5(3)				3A,B	4A,B			
		ST.5(4)				3A,B	4A,B			
System compatibility	Communication compatibility	SY.1(1)	0	1	2				2	2
		SY.1(2)	0	1	2				2	2
		SY.1(3)	0	1	2				2	2
		SY.1(4)	0	1	2				2	2
	Data compatibility	SY.2(1)	0	1	2				2	2
		SY.2(2)	0	1	2				2	2
		SY.2(3)	0	1	2				2	2
	Hardware compatibility	SY.3(1)	0	1						
		SY.3(2)	0	1					1	0
		SY.3(3)	0	1						
		SY.3(4)	0	1					1	0
		SY.3(5)	0	1						
		SY.3(6)	0	1					1	0
	Software compatibility	SY.4(1)	0	1	2				2	2
		SY.4(2)	0	1	2				2	2
		SY.4(3)	0	1	2				2	2
	Documentation for other system	SY.5(1)	0	1						
Traceability	Cross reference	TC.1(1)		1	2	3A				
		TC.1(2)				3A				
Training	Training checklist	TN.1(1)	0	1		3A			3A	3A
		TN.1(2)	0	1		3A			3A	3A
		TN.1(3)	0	1		3A			3A	3A
		TN.1(4)	0	1		3A			3A	3A
Virtuality	System data independence	VR.1(1)	0	1	2				2	2
Visibility	Unit testing	VS.1(1)				3A,B	4A,B			
		VS.1(2)				3A,B	4A,B			
	Integration testing	VS.2(1)				3A		3A		
	CSCI testing	VS.3(1)				3A			3A	
		VS.3(2)				3A			3A	
		VS.3(3)				3A			3A	

WORKSHEET PREPARATION INSTRUCTIONS

SECTION A - GENERAL INFORMATION

1. Enter the name and contract number of the project.
2. Enter the date the worksheet is prepared (month, day, year).
3. Enter the name of the person responsible for completing the worksheet.
4. Enter the name of the entity to which the worksheet is applied.
5. Enter the number and name of all documentation which is used as source material for answering the metric questions in Section B.
6. Enter comments reflecting the inspector's observations on product quality and any additional information regarding specific metric questions. Attach additional sheets, as necessary.

SECTION B - METRIC QUESTIONS

Answer all applicable metric questions by circling the appropriate answer (Y = yes, N = no, N/A = not applicable) or by entering the appropriate value. A glossary of terms is provided at the end of this section.

METRIC WORKSHEET 0 SYSTEM LEVEL

SECTION A - GENERAL INFORMATION

1. PROJECT ______________________
2. DATE ______________________
3. INSPECTOR ______________________
4. PRODUCT ______________________
5. SOURCE DOCUMENTATION:

____
____
____
____
____
____
____

INSPECTOR COMMENTS:

0-2

SECTION B - METRIC QUESTIONS

AC.1(1) Have accuracy requirements been established for all applicable missions/ operations? Y N N/A

AC.1(2) Have all accuracy requirements been budgeted to individual functions? Y N N/A

AC.1(3) Are there quantitative accuracy requirements for all applicable inputs associated with each applicable function (e.g., mission-critical function)? Y N N/A

AC.1(4) Are there quantitative accuracy requirements for all applicable outputs associated with each applicable function (e.g., mission-critical function)? Y N N/A

AC.1(5) Are there quantitative accuracy requirements for all applicable constants associated with each applicable function (e.g., mission-critical function)? Y N N/A

AC.1(6) Do the existing math library routines which are planned for use provide enough precision to support accuracy objectives? Y N N/A

AM.1(1) a. How many instances are there of different processes (or functions, sub-functions) which are required to be executed at the same time (i.e., concurrent processing)? ____ N/A

b. How many instances of concurrent processing are required to be centrally controlled? ____ N/A

c. Calculate b/a and enter score. ____ N/A

AM.1(2) a. How many error conditions are required to be recognized (identified)? ____ N/A

b. How many recognized error conditions require recovery or repair? ____ N/A

c. Calculate b/a and enter score. ____ N/A

AM.1(3) Is there a standard for handling recognized errors such that all error conditions are passed to the calling function or software element? Y N N/A

AM.1(4)
a. How many instances of the same process (or function, subfunction) being required to execute more than once for comparison purposes (e.g., polling of parallel or redundant processing results)? ____ N/A

b. How many instances of parallel/redundant processing are required to be centrally controlled? ____ N/A

c. Calculate b/a and enter score. ____ N/A

AM.2(1) Are error tolerances specified for all applicable external input data (e.g., range of numerical values, legal combinations of alphanumerical values)? Y N N/A

AM.3(1) Are there requirements for detection of and/or recovery from all computational failures? Y N N/A

AM.4(1) Are there requirements to recover from all detected hardware faults (e.g., arithmetic faults, power failure, clock interrupt)? Y N N/A

AM.5(1) Are there requirements to recover from all I/O device errors? Y N N/A

AM.6(1) Are there requirements to recover from all communication transmission errors? Y N N/A

AM.7(1) Are there requirements to recover from all failures to communicate with other nodes or other systems? Y N N/A

AM.7(2) Are there requirements to periodically check all adjacent nodes or interoperating systems for operational status? Y N N/A

AM.7(3) Are there requirements to provide a strategy for alternate routing of messages? Y N N/A

AP.1(1)	Is there a requirement to limit specific references to the data base management scheme (e.g., all data calls for data base information processed through an executive)?	Y N N/A
AP.2(2)	Is there a standard for commenting all global data within a software unit to show where data is derived, the data's composition, and how the data is used?	Y N N/A
AP.2(4)	Is there a standard for commenting all parameter input/output and local variables within a software unit which includes requirements identifying the data's composition and use?	Y N N/A
AP.3(1)	Is there a requirement to localize specific references to computer architecture (e.g., specific device references localized to the executive rather than the application software)?	Y N N/A
AP.4(1)	Is there a requirement to avoid or to limit the use of microcode instruction statements?	Y N N/A
AP.5(1)	Is there a requirement to develop functional processing algorithms such that they are not unique to this system's application (e.g., can be used in a similar application with, at most, minimum tailoring)?	Y N N/A
AT.1(2)	Are there requirements for spare memory storage capacity?	Y N N/A
AT.1(3)	Are there requirements for spare auxiliary storage capacity?	Y N N/A
AT.2(3)	Are there requirements for spare processing capacity (time)?	Y N N/A
AT.3(1)	Are there requirements for spare I/O channel capacity (time)?	Y N N/A
AT.3(2)	Are there requirements for spare communication channel capacity (time)?	Y N N/A
AT.4(1)	Are there requirements for interface compatibility among all the processors, communication links, memory devices, and peripherals?	Y N N/A

AT.4(2) Is documentation available which describes the results of any previous engineering studies such as tradeoff studies, feasibility studies, risk analyses, and requirements definitions? Y | N | N/A

AT.4(3) Is documentation available which describes new or emerging software-related disciplines which may affect the scope of the software requirements or the software implementation techniques (e.g., voice recognition using artificial intelligence techniques)? Y | N | N/A

AU.1(1) Are all processes and functions partitioned to be logically complete and self-contained so as to minimize interface complexity? Y | N | N/A

AU.2(1) Are there requirements for each operational CPU/system to have a separate power source? Y | N | N/A

AU.2(2) Are there requirements for the executive software to perform testing of its own operation and of the communication links, memory devices, and peripheral devices? Y | N | N/A

CL.1(1) Are there requirements for communication with other systems? Y | N | N/A

CL.1(2) Is there a requirement for a protocol standard to control all network communication? Y | N | N/A

CL.1(3) Is network processing control part of the network protocol standard? Y | N | N/A

CL.1(4) Is user session control part of the network protocol standard? Y | N | N/A

CL.1(5) Is communication routing part of the network protocol standard? Y | N | N/A

CL.1(6) Is uniform message handling (e.g., synchronization, message decoding) part of the network protocol standard? Y | N | N/A

CL.1(7) a. How many functions receive inputs from other systems? ____ N/A

b. Calculate 1/a and enter score. ____ N/A

0-6

CL.1(8) a. How many functions transmit outputs to other systems? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

CL.1(9) a. How many other systems must respond correctly to successfully complete synchronization? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

CL.1(10) Does the time to perform successful synchronization impose constraints upon system computation or response time (e.g., result in user wait time of more than several seconds)? Y | N | N/A

CL.1(11) Is the system free from time-critical constraints with respect to external communication (e.g., data freshness)? Y | N | N/A

CL.1(12) a. How many other systems is this system required to interface with? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

CL.1(13) Is there a general description of how the computer system appears to the users and how the users interact with the computer system (e.g., operational concept document)? Y | N | N/A

CL.1(14) Is there a complete and definitive set of operating procedures for using this system? Y | N | N/A

CL.2(1) Are there requirements for a standard to establish common representations of data for uniform communication with other systems? Y | N | N/A

CL.2(2) a. How many functions perform data translations? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

/0'✓

CL.2(3) Is there a requirement to receive all input data from other systems in common formats (e.g., common formats for data positioning, data packing, block transmission)? Y N N/A

CL.2(4) a. How many different formats are used for input data from other systems (e.g., formats for data positioning, data packing, block transmission)? ___ N/A

b. Calculate 1/(1+a) and enter score. ___ N/A

CL.2(5) Is there a requirement to output all data to other systems in common formats (e.g., common formats for data positioning, data packing, block transmission)? Y N N/A

CL.2(6) a. How many different formats are used for output data to other systems (e.g., formats for data positioning, data packing, block transmission)? ___ N/A

b. Calculate 1/(1+a) and enter score. ___ N/A

CL.2(7) a. How many different types of input records are received from other systems? ___ N/A

b. How many types of input records received from other systems contain tags identifying the type of data they contain? ___ N/A

c. Calculate b/a and enter score. ___ N/A

CL.2(8) a. How many different types of output records are transmitted to other systems? ___ N/A

b. How many types of output records transmitted to other systems contain tags identifying the type of data they contain? ___ N/A

c. Calculate b/a and enter score. ___ N/A

CL.3(1) Has a common technical vocabulary with equivalent definitions been established for use with this system and for use with all interoperating systems (e.g., definition and use of data item, block, record)? Y N N/A

CP.1(1) Are all inputs, processing, and outputs clearly and precisely defined? Y N N/A

CP.1(2) a. How many data references are identified? N/A

b. How many identified data references are documented with regard to source, meaning, and format? N/A

c. Calculate b/a and enter score. N/A

CP.1(3) a. How many data items are defined (i.e., documented with regard to source, meaning, and format? N/A

b. How many data items are referenced? N/A

c. Calculate b/a and enter score. N/A

CP.1(5) Have all defined functions (i.e., documented with regard to source, meaning and format) been referenced? Y N N/A

CP.1(6) Have all system functions been allocated to configuration items (i.e., CSCI's, HWCI's)? Y N N/A

CP.1(7) Have all referenced functions been defined (i.e., documented with precise inputs, processing, and output requirements)? Y N N/A

CP.1(8) Are the processing flows (algorithms) and all decision points (conditions and alternate paths) in the flows described for all functions? Y N N/A

CP.1(11) a. How many software problem reports have been recorded, to date? N/A

b. How many recorded software problem reports have been closed (resolved), to date? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

CS.1(1) Is there a requirement to standardize all design representations (e.g., representations for control flow, data flow)? Y N N/A

CS.1(2) Is there a requirement to standardize the calling sequence protocol between software units? Y N N/A

CS.1(3) Is there a requirement to standardize the external I/O protocol and format for all software units? Y N N/A

CS.1(4) Is there a requirement to standardize error handling for all software units? Y N N/A

CS.1(5) Do all references to the same function use a single, unique name? Y N N/A

CS.2(1) Is there a requirement to standardize all data representation in the design? Y N N/A

CS.2(2) Is there a requirement to standardize the naming of all data? Y N N/A

CS.2(3) Is there a requirement to standardize the definition and use of global variables? Y N N/A

CS.2(4) Are there requirements to establish consistency and concurrency of multiple copies (e.g., copies at different nodes) of the same software or data base version? Y N N/A

CS.2(5) Are there requirements to verify consistency and concurrency of multiple copies (e.g., copies at different nodes) of the same software or data base version? Y N N/A

CS.2(6) Do all references to the same data use a single, unique name? Y N N/A

DI.1(1) Is a graphic portrayal (e.g., figures, diagrams, tables) provided which identifies all software functions and functional interfaces? Y N N/A

DI.1(2)	Is a graphic portrayal provided which identifies all different types of system-level information and the information flow within the system?	Y N N/A
DI.1(3)	Are there requirements for the organization and distribution of information within the system (e.g., information distributed across nodes or among storage devices)?	Y N N/A
DI.1(4)	Are there requirements for file/library accessibility from each node?	Y N N/A
DI.1(5)	Are there requirements for providing alternate processing sources within the system (e.g., multiple processors, alternate node)?	Y N N/A
DI.1(6)	Are there requirements to distribute all mission-critical functions over redundant elements or nodes?	Y N N/A
DI.1(7)	Are there requirements to distribute control functions across different nodes/elements so as to ensure system operation under anomalous conditions?	Y N N/A
DI.1(8)	Do the requirements allow for implementing functions across several physical structures (i.e., function and physical structure are not necessarily the same)?	Y N N/A
DI.1(9)	Are there requirements regarding the number of nodes that can be removed from an operational system such that each remaining node still maintains its capability to communicate with all other nodes?	Y N N/A
DO.1(1)	Are current versions of all software documentation related to the project free from access control (i.e., any member of the current project or other projects may access a copy of any document)?	Y N N/A
DO.2(1)	Is all the documentation structured and written clearly and simply such that procedures, functions, algorithms, etc. can be easily understood?	Y N N/A
DO.2(2)	Does the requirements/design documentation clearly depict control and data flow (e.g., graphic portrayal with accompanying explanations or PDL)?	Y N N/A

DO.2(3) Does all documentation contain an indexing scheme which facilitates quickly locating and accessing various information in the document (e.g., hierarchical structured table of contents, inserted tabs)? Y N N/A

DO.2(4) Do the software specifications and design documentation have separate volumes or separations within a single volume based on system functions or software functions? Y N N/A

DO.2(5) Does the documentation completely characterize the operational capabilities of the software (e.g., identify all the performance parameters and limitations)? Y N N/A

DO.2(6) Does the documentation contain comprehensive descriptions of all system/software functions including functional processing, functional algorithms, and functional interfaces? Y N N/A

EC.1(1) Have performance requirements and limitations for system communication efficiency been specified for each system function? Y N N/A

EP.1(1) Have performance requirements and limitations for processing efficiency been specified for each system function (e.g., flow time for process, execution time)? Y N N/A

EP.1(3) Is there a requirement to use an optimizing compiler or to code in assembly language to optimize processing efficiency? Y N N/A

EP.1(5) Is memory management of the system free from requirements for overlays? Y N N/A

EP.2(1) Have performance requirements and limitations for storing data to efficiently process it been specified for each system function? Y N N/A

EP.2(2) Are there requirements to efficiently process stored information (e.g., rapidly update files, buffers, etc.)? Y N N/A

EP.2(3) Does the source code language(s) enable variable initialization when the variable is declared? Y N N/A

ES.1(1) Have performance requirements and limitations for storing data to efficiently utilize primary and secondary storage been specified for each system function (e.g., data packing, dynamic memory management)? Y N N/A

ES.1(2) Does the memory management of the system incorporate virtual storage? Y N N/A

ES.1(5) Does the memory management of the system incorporate dynamic reallocation of physical memory space during execution (dynamic memory management)? Y N N/A

ES.1(7) Is there a requirement to use an optimizing compiler or to code in assembly language to optimize storage efficiency? Y N N/A

ES.1(8) Are there requirements to avoid redundant storage of files and libraries? Y N N/A

FO.1(1)

a. How many functions in this system? N/A

b. How many system functions are duplicated in interoperating systems? N/A

c. Calculate 1-(b/a) and enter score. N/A

FO.1(2)

a. How many duplicated function sets exist between this system and interoperating systems (i.e., the same function is performed in this system and in an interoperating system)? N/A

b. In how many instances of the duplicated function sets will duplicated functions be deleted (i.e., leaving the responsibility for performing the function with one system)? N/A

c. Calculate 1-(b/a) and enter score. N/A

FO.1(3)

a. How many duplicated function sets exist between this system and interoperating systems (i.e., the same function is performed in this system and in an interoperating system)? N/A

b. How many of the duplicated function sets require synchronization of the functions within the set? N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FO.1(4) a. How many duplicated function sets exist between this system and interoperating systems (i.e., the same function is performed in this system and in an interoperating system)? [ ] N/A

b. How many of the duplicated function sets require redundancy management techniques/logic to enable system interoperability? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FS.2(1) Are there requirements to construct functions in such a way so as to facilitate their use in other similar system applications? Y N N/A

FS.2(2) a. How many system functions? [ ] N/A

b. How many system functions are constructed in such a way so as to facilitate their use in other similar system applications? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FS.2(3) Are all inputs documented as to the specific use and limitations of the data? Y N N/A

FS.2(4) Are all input/output formats specified and documented? Y N N/A

FS.2(5) Are all outputs documented as to the specific use and interpretation of the data Y N N/A

FS.2(6) a. How many system functions? [ ] N/A

b. How many system functions will likely satisfy the requirements of other similar applications [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FS.3(1) Are there requirements to provide the user options for computation and output (e.g., selection of type of coordinate system, output media, format)? Y N N/A

FS.3(2) Are there requirements to enable modification of the resources allocated to functions (e.g., changing the amount of memory work space for a function)? Y N N/A

FS.3(3)
a. How many functions are typically performed by a system for this application? N/A

b. How many functions does this system perform? N/A

c. Calculate b/a and enter score. (Note that if b/a is greater than one, enter one.) N/A

ID.1(2) Is there a requirement to use a standard subset of the implementation language(s)? Y N N/A

ID.1(3) Has a standard subset of the implementation language(s) been established for coding? Y N N/A

ID.2(1) Are the same version and dialect of the implementation language(s) supported on other machines? Y N N/A

MO.1(1) Are there requirements to develop all software functions and software elements according to structured design techniques (e.g., top-down design)? Y N N/A

MO.1(2) Are all software functions and CSCI's developed according to structured design techniques? Y N N/A

MO.2(1) Are there requirements regarding the relationships among software entities (i.e., types of coupling allowed among software functions, CSCI's, CSC's and units) (e.g., requirements to minimize content, common and external coupling among software entities)? Y N N/A

MO.2(4) Are there requirements regarding the relationship between the elements within a software entity (i.e., cohesion value for software functions, CSCI's, CSC's, and units) (e.g., all software entities are required to reflect an average cohesion value of 0.6 or greater)? Y | N | N/A

OP.1(1) Have the operating characteristics of the system been specified (i.e., the normal and alternate procedures and actions performed by the system? Y | N | N/A

OP.1(2) Are all the errors specified which are to be reported to the operator/user? Y | N | N/A

OP.1(3) Are the required operator/user responses specified for all reported errors? Y | N | N/A

OP.1(4) Are there requirements to provide the operator with the capability to interrupt system operation, obtain operational status, save and enter data, and continue processing? Y | N | N/A

OP.1(5) a. How many operations/responses are performed by the operator for a typical mission/job? [ ] N/A

b. Calculate 1/(1+a) and enter score. [ ] N/A

OP.1(6) Are there requirements to specify the procedures for setting up a mission/job and completing it? Y | N | N/A

OP.1(8) Are there requirements to maintain a hard copy log of all operator interactions with the system? Y | N | N/A

OP.1(10) Are there requirements to provide simple and consistent operator messages and require simple and consistent operator responses (i.e., minimize the number of operator message and response formats; use the same format types throughout the system)? Y | N | N/A

OP.1(11) Are there requirements to report all access violations to the operator? Y | N | N/A

OP.1(12)	Are there requirements specifying the appropriate response(s) (by the operator, the system/software, or both) for all access violations?	Y N N/A
OP.1(13)	Are there requirements to provide the operator/system the capability to obtain specific system (or network) resource status information and to reallocate resources?	Y N N/A
OP.1(14)	Are there requirements to provide the operator/user the capability to select different nodes for different types of processing or for retrieval of information?	Y N N/A
OP.1(15)	Are there requirements to provide the operator/user the capability to manipulate data regardless of the data's location in the system?	Y N N/A
OP.1(16)	Are there requirements to make system implementation details transparent to the user (e.g., the user can access a file without knowing its location in the system/network)?	Y N N/A
OP.2(1)	a. What is the total number of different user input parameters?	N/A
	b. How many input parameters have default values?	N/A
	c. Calculate b/a and enter score.	N/A
OP.2(4)	Are there requirements to enable the user to review and modify all input data prior to execution?	Y N N/A
OP.2(5)	Are there requirements to terminate all user-input data by explicitly defined logical end of input?	Y N N/A
OP.2(6)	Are there requirements to provide the user options for input media (e.g., terminal, tape drive, card reader)?	Y N N/A
OP.3(1)	Are there requirements to provide the user with output control (e.g., choosing specific outputs, output media, output formats, amount of output?	Y N N/A

OP.3(2)	Is there a requirement for all outputs to the user to have unique, descriptive labels for identifying data?	Y N N/A
OP.3(3)	Is there a requirement to provide all outputs to the user in user-oriented measurement units?	Y N N/A
OP.3(6)	Are there requirements for all error messages to clearly identify the nature of the error to the user?	Y N N/A
OP.3(7)	Are there requirements to provide the user with options for output media?	Y N N/A
OP.3(8)	Are there requirements to establish a standard (common) user command language for network information and data access?	Y N N/A
RE.1(1)	Are there requirements to ensure communication paths to all remaining nodes/communication links in the event of a failure of one node/link?	Y N N/A
RE.1(2)	Are there requirements for maintaining the integrity of all data values following the occurrence of anomalous conditions?	Y N N/A
RE.1(3)	Are there requirements to enable all disconnected nodes to rejoin the network after recovery, such that the processing functions of the system are not interrupted?	Y N N/A
RE.1(4)	Are there requirements to replicate all critical data in the system at two or more distinct nodes?	Y N N/A
SD.2(1)	Has the specific standard been established that each unit prologue contain the unit's function, author, version number, version date, inputs, outputs, algorithms, assumptions and limitations.	Y N N/A
SD.2(2)	Has a standard been established for the identification and placement of comments in the unit?	Y N N/A
SD.3(5)	Has a standard format for the structure of units been established?	Y N N/A

0-18

SI.1(1)	Are there diagrams identifying all functions in a structured fashion (e.g., top-down hierarchical)?	Y N N/A
SI.1(8)	Are there requirements for a programming standard?	Y N N/A
SI.1(9)	Has a programming standard been established?	Y N N/A
SI.2(1)	Are there requirements to use a structured language or preprocessor to implement the software?	Y N N/A
SI.4(13)	Are there requirements for a programming standard?	Y N N/A
SS.1(1)	Are there requirements to control user input/output access in the system (e.g., user access is limited by identification and password checking)?	Y N N/A
SS.1(2)	Are there requirements to control data access in the system?	Y N N/A
SS.1(3)	Are there requirements to control the scope of task operations during execution (e.g., tasks cannot invoke other tasks, access system registers, or use privileged commands)?	Y N N/A
SS.1(4)	Are there requirements to control access to the network?	Y N N/A
SS.2(1)	Are there requirements to record and report all access to the system (e.g., record terminal and processor linkage, data file access, and jobs run information)?	Y N N/A
SS.2(2)	Are there requirements to immediately indicate and identify all access violations?	Y N N/A
ST.3(1)	Are there requirements to isolate I/O functions from computational functions?	Y N N/A
ST.3(2)	Are I/O functions isolated from computational functions?	Y N N/A

SY.1(1)	Are there requirements for the I/O transmission rates of this system to be the same as the interoperating system(s)?	Y N N/A
SY.1(2)	Are there requirements for this system to use the same communication protocol as the interoperating system(s)?	Y N N/A
SY.1(3)	Are there requirements for common interpretation of the content in all messages sent from and received by this system and by the interoperating system(s) (e.g., all variables in the message have the same meaning)?	Y N N/A
SY.1(4)	Are there requirements for this system to use the same structure and sequence for message contents as the interoperating system(s) (e.g., all real variables are 16 bits in length; all real coordinates are ordered XCOORD, YCOORD, ZCOORD)?	Y N N/A
SY.2(1)	Are there requirements for this system to use the same data format as the interoperating system(s) (e.g., all characters are represented in ASCII format)?	Y N N/A
SY.2(2)	Are there requirements for this system to establish the same data base structure as the interoperating system(s) (e.g., all systems use a relational data base containing similar information)?	Y N N/A
SY.2(3)	Are there requirements for this system to provide the same data base access techniques as the interoperating system(s)?	Y N N/A
SY.3(1)	Are there requirements for this system to use the same word length as the interoperating system(s)?	Y N N/A
SY.3(2)	Does this system use the same word length as the interoperating system(s)?	Y N N/A
SY.3(3)	Are there requirements for this system to use the same interrupt structure as the interoperating system(s)?	Y N N/A
SY.3(4)	Does this system use the same interrupt structure as the interoperating system(s)?	Y N N/A

0-20

SY.3(5)	Are there requirements for this system to use the same instruction set as the interoperating system(s)?	Y N N/A
SY.3(6)	Does this system use the same instruction set as the interoperating system(s)?	Y N N/A
SY.4(1)	Are there requirements for this system to use the same source code language(s) as the interoperating system(s)?	Y N N/A
SY.4(2)	Are there requirements for this system to use the same operating system as the interoperating system(s)?	Y N N/A
SY.4(3)	Are there requirements for this system to use the same support software as the interoperating system(s)?	Y N N/A
SY.5(1)	Is documentation available from the interoperating system(s) that enables interoperability requirements to be established for this system (e.g., documentation is up-to-date, complete, and clearly organized)?	Y N N/A
TN.1(1)	Are there requirements to provide lesson plans and training materials for operators, and users, and maintainers of the system?	Y N N/A
TN.1(2)	Are there requirements to provide realistic simulation exercises for the system?	Y N N/A
TN.1(3)	Are there requirements, to provide "help" information and diagnostic information for the operator, end user, and maintainer of the system (e.g., provide an on-line list of legal commands or a list of the sequential steps in a process)?	Y N N/A
TN.1(4)	Are there requirements to provide selectable levels of aid and guidance for system users of different degrees of expertise?	Y N N/A
VR.1(1)	Are there requirements to make system implementation details transparent to the user (e.g., the user can create a file without specifying its location in the system/network)?	Y N N/A

0-21

GLOSSARY

Anomalous Condition: An event resulting in a deviation from the normal operating environment or procedures.

Cohesion Value: The type of relationship that exists among the elements of each software entity (Function, CSCI, Unit). The following are relative values for seven types of cohesion:

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohension.

1) Coincidental
   - No meaningful relationships among the elements of an entity.
   - Difficult to describe the module's function(s).
2) Logical
   - Entity performs (at each invocation) one of a class of related functions (e.g., "edit all data").
   - Entity performs more than one function.
3) Classical
   - Entity performs one of a class of functions that are related in time (Program procedure).
   - Entity performs more than one function.
4) Procedural
   - Entity performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).
5) Communicational
   - Entity has procedural strength; in addition, all of the elements "communicate"

Glossary (Continued)

with one another (e.g., reference same data or pass data among themselves).

- All functions use the same data.

6) Informational

- Entity performs multiple functions where the functions (entry points in the module) deal with a single data structure.
- Physical packaging together of two or more entities having functional strength.
- All functions use the same data.

7) Functional

- All entity elements are related to the performance of a single function.

Command Language: The set of instructions used to invoke specific operations in a computer software subsystem/program.

Communication Channel: The pathways along which data/messages are communicated to the various system components or nodes (i.e., other computer, data storage units, special processors, etc.).

Coupling: The type of relationship that exists between two software entities (Functions, CSCIs, Units). In achieving a highly modular design it is essential to minimize the relationships among software entities. The goal is to design software entities with low coupling. The scae of coupling from worst to best is: (1) Content Coupling, 2) Common Coupling, 3) External Coupling, 4) Control Coupling, 5) Stamp Coupling, and 6) Data Coupling.

1) Content Coupling - One software entity references the contents of another software entity.

2) Common Coupling - Software entities reference a shared global data structure.

3) External Coupling - Software entities reference the same externally declared symbol.

Glossary (Continued)

4) Control Coupling - One software entity passes control elements as arguments to another software entity.

5) Stamp Coupling - Two software entities reference the same data structure, which is not global.

6) Data Coupling - One software entity calls another and the software entities are not coupled as defined above (in 1 through 5).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Format: The positioning, packing or organization of the order that the data appears in.

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Record: A structured grouping of related data elements for the purpose of storage or transmission.

Data Reference: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Database Management Scheme: The methods and commands used to access or operate the database management software system.

Design Representation: A formal statement of the details or organization of a design using one of a number of design representation methodologies, such as, Flow Charts, HIPO Charts, PDL, etc.

Error Analysis: A study to determine the minimum acceptable level of performance and precision, allocates the accuracy requirements for the individual functions to be performed by the system.

I/O Channel: The pathways along which data/messages are communicated to the various user-oriented peripherals in the system (e.g., CRT, Printer).

Glossary (Continued)

Microcode Instruction: A low-level computer instruction specifying a single machine operation.

Mission-Critical Function: A feature essential to fulfilling the desired objectives of the system.

Multiple Transfer Index Parameters: A value used to select a variation in the order of code execution (i.e., case statement, program switch, etc.).

Network: A system of computers, terminals, and data bases that are linked/interconnected with the use of communication lines.

Node: The points at which subsidiary parts originate or connect to a system containing interconnected system parts or devices.

Range-Test: A test performed to validate the object of interest over the complete spectrum of applicable values.

Subscript Value: A value used to reference an entity from a group of related objects (i.e., table index, array index, etc.).

Synchronization: The process of ensuring that two or more components of a system are ready and capable of communicating with one another.

Virtual Storage: The storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

WORKSHEET PREPARATION INSTRUCTIONS

SECTION A - GENERAL INFORMATION

1. Enter the name and contract number of the project.
2. Enter the date the worksheet is prepared (month, day, year).
3. Enter the name of the person responsible for completing the worksheet.
4. Enter the name of the entity to which the worksheet is applied.
5. Enter the number and name of all documentation which is used as source material for answering the metric questions in Section B.
6. Enter comments reflecting the inspector's observations on product quality and any additional information regarding specific metric questions. Attach additional sheets, as necessary.

SECTION B - METRIC QUESTIONS

Answer all applicable metric questions by circling the appropriate answer (Y = yes, N = no, N/A = not applicable) or by entering the appropriate value. A glossary of terms is provided at the end of this section.

METRIC WORKSHEET 1 CSCI LEVEL

SECTION A - GENERAL INFORMATION

1. PROJECT ______________________
2. DATE ______________________
3. INSPECTOR ______________________
4. PRODUCT ______________________
5. SOURCE DOCUMENTATION:

____
____
____
____
____
____
____

6. INSPECTOR COMMENTS:

1-2

/ 4c

SECTION B - METRIC QUESTIONS

AC.1(3) Are there quantitative accuracy requirements for all applicable inputs associated with each applicable function (e.g., mission-critical function)? Y | N | N/A

AC.1(4) Are there quantitative accuracy requirements for all applicable outputs associated with each applicable function (e.g., mission-critical function)? Y | N | N/A

AC.1(5) Are there quantitative accuracy requirements for all applicable constants associated with each applicable function (e.g., mission-critical function)? Y | N | N/A

AC.1(6) Do the existing math library routines which are planned for use provide enough precision to support accuracy objectives? Y | N | N/A

AM.1(1) a. How many instances are there of different processes (or functions, sub-functions) which are required to be executed at the same time (i.e., concurrent processing)? ____ | N/A

b. How many instances of concurrent processing are required to be centrally controlled? ____ | N/A

c. Calculate b/a and enter score. ____ | N/A

AM.1(2) a. How many error conditions are required to be recognized (identified)? ____ | N/A

b. How many recognized error conditions require recovery or repair? ____ | N/A

c. Calculate b/a and enter score. ____ | N/A

AM.1(3) Is there a standard for handling recognized errors such that all error conditions are passed to the calling function or software element? Y | N | N/A

AM.1(4) a. How many instances of the same process (or function, subfunction) being required to execute more than once for comparison purposes (e.g., polling of parallel or redundant processing results)? [ ] N/A

b. How many instances of parallel/redundant processing are required to be centrally controlled? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

AM.2(1) Are error tolerances specified for all applicable external input data (e.g., range of numerical values, legal combinations of alphanumerical values)? Y N N/A

AM.3(1) Are there requirements for recovery from all computational failures? Y N N/A

AM.3(2) Are there requirements to range test all critical (e.g., supporting a mission-critical function) loop and multiple transfer index parameters before use? Y N N/A

AM.3(3) Are there requirements to range test all critical (e.g., supporting a mission-critical function) subscript values before use? Y N N/A

AM.3(4) Are there requirements to check all critical output data (e.g., data supporting a mission critical system function) before final outputting? Y N N/A

AM.4(1) Are there requirements for recovery from all detected hardware faults (e.g., arithmetic faults, power failure, clock interrupt)? Y N N/A

AM.5(1) Are there requirements for recovery from all I/O device errors? Y N N/A

AM.6(1) Are there requirements for recovery from all communication transmission errors? Y N N/A

AM.7(1) Are there requirements for recovery from all failures to communicate with other nodes or other systems? Y N N/A

1-4

AM.7(2)	Are there requirements to periodically check all adjacent nodes or interoperating systems for operational status?		Y N N/A
AM.7(3)	Are there requirements to provide a strategy for alternate routing of messages?		Y N N/A
AP.1(1)	Is there a requirement to limit specific references to the data base management scheme (e.g., all data calls for data base information are processed through an executive)?		Y N N/A
AP.2(2)	Is there a standard for commenting all global data within a software unit to show where data is derived, the data's composition, and how the data is used?		Y N N/A
AP.2(4)	Is there a standard for commenting all parameter input and output and local variables within a software unit which includes requirements for identifying the data's composition and use?		Y N N/A
AP.3(1)	Is there a requirement to localize specific references to computer architecture (e.g., specific device references localized to the executive rather than the application software)?		Y N N/A
AP.4(1)	Is there a requirement to avoid or to limit the use of microcode instruction statements?		Y N N/A
AP.5(1)	a.	How many functions?	___ N/A
	b.	How many functions are not unique to this application (e.g., can be used in a similar application with, at most, minimum tailoring)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
AT.1(2)	Are there requirements for spare memory storage capacity?		Y N N/A
AT.1(3)	Are there requirements for spare auxiliary storage capacity?		Y N N/A

1-5

METRIC WORKSHEET 1 CSCI LEVEL

AT.2(3)	Are there requirements for spare processing capacity (time)?	Y N N/A
AT.3(1)	Are there requirements for spare I/O channel capacity (time)?	Y N N/A
AT.3(2)	Are there requirements for spare communication channel capacity (time)?	Y N N/A
AT.4(1)	Are there requirements for interface compatibility among all the processors, communication links, memory devices, and peripherals?	Y N N/A
AT.4(2)	Is documentation available which describes the results of any previous engineering studies such as tradeoff studies, feasibility studies, risk analyses, and requirements definitions?	Y N N/A
AT.4(3)	Is documentation available which describes new or emerging software-related disciplines which may affect the scope of the software requirements or the software implementation techniques (e.g., voice recognition using artificial intelligence techniques)?	Y N N/A
AU.1(1)	Are all processes and functions partitioned to be logically complete and self-contained so as to minimize interface complexity?	Y N N/A
AU.2(1)	Does each operational CPU/system have a separate power source?	Y N N/A
AU.2(2)	Are there requirements for the executive software to perform testing of its own operation and of the communication links, memory devices, and peripheral devices?	Y N N/A
CL.1(1)	Are there requirements for communication with other systems?	Y N N/A
CL.1(2)	Is there a requirement for a protocol standard to control all network communication?	Y N N/A
CL.1(3)	Is network processing control part of the network protocol standard?	Y N N/A
CL.1(4)	Is user session control part of the network protocol standard?	Y N N/A

1-6

CL.1(5) Is communication routing part of the network protocol standard? Y N N/A

CL.1(6) Is uniform message handling (e.g., synchronization, message decoding) part of the network protocol standard? Y N N/A

CL.1(7) a. How many functions receive inputs from other systems? ____ N/A

b. Calculate 1/a and enter score. ____ N/A

CL.1(8) a. How many functions transmit outputs to other systems? ____ N/A

b. Calculate 1/a and enter score. ____ N/A

CL.1(9) a. How many other systems must respond correctly to successfully complete synchronization? ____ N/A

b. Calculate 1/a and enter score. ____ N/A

CL.1(10) Does the time to perform successful synchronization impose constraints upon system computation or response time (e.g., result in user wait time of more than several seconds)? Y N N/A

CL.1(11) Is the CSCI free from time-critical constraints with respect to external communication (e.g., data freshness)? Y N N/A

CL.1(12) a. How many other systems is this CSCI required to interface with? ____ N/A

b. Calculate 1/a and enter score. ____ N/A

CL.1(13) Is there a general description of how the computer system appears to the users and how the users interact with the computer system (e.g., operational concept document)? Y N N/A

CL.1(14) Is there a complete and definitive set of operating procedures for using this system? Y N N/A

CL.2(1) Has a standard been established for common representations of data and/or for translations between representations of data for uniform communication with other systems? Y N N/A

CL.2(2) a. How many functions perform data translations? ____ N/A

b. Calculate 1/a and enter score. ____ N/A

CL.2(3) Is there a requirement to receive all input data from other systems in common formats (e.g., common formats for data positioning, data packing, block transmission)? Y N N/A

CL.2(4) a. How many different formats are used for input data from other systems (e.g., formats for data positioning, data packing, block transmission)? ____ N/A

b. Calculate 1/(1+a) and enter score. ____ N/A

CL.2(5) Is there a requirement to output all data to other systems in common formats (e.g., common formats for data positioning, data packing, block transmission)? Y N N/A

CL.2(6) a. How many different formats are used for output data to other systems (e.g., formats for data positioning, data packing, block transmission)? ____ N/A

b. Calculate 1/(1+a) and enter score. ____ N/A

CL.2(7) a. How many different types of input records are received from other systems? ____ N/A

b. How many types of input records received from other systems contain tags identifying the type of data they contain? ____ N/A

	c.	Calculate b/a and enter score.	[ ] N/A
CL.2(8)	a.	How many different types of output records are transmitted to other systems?	[ ] N/A
	b.	How many types of output record transmitted to other systems contain tags identifying the type of data they contain?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
CL.3(1)		Has a common technical vocabulary with equivalent definitions been established for use with this system and for use with all interoperating systems (e.g., definition and use of data item, block, record)?	Y N N/A
CP.1(1)		Are all inputs, processing, and outputs clearly and precisely defined?	Y N N/A
CP.1(2)	a.	How many data references are identified?	[ ] N/A
	b.	How many identified data references are documented with regard to source, meaning, and format?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
CP.1(3)	a.	How many data items are defined (i.e., documented with regard to source, meaning, and format)?	[ ] N/A
	b.	How many defined data items are referenced?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
CP.1(5)		Have all defined functions (i.e., documented with regard to source, meaning, and format) been referenced?	Y N N/A
CP.1(6)		Have all system functions allocated to this CSCI been allocated to software functions within this CSCI?	Y N N/A

CP.1(7) Have all referenced functions been defined (i.e., documented with precise inputs, processing, and output requirements)? Y N N/A

CP.1(8) Are the processing flows (algorithms) and all decision points (conditions and alternate paths) in the flows described for all functions? Y N N/A

CP.1(11) a. How many software problem reports have been recorded, to date? N/A

b. How many recorded software problem reports have been closed (resolved), to date? N/A

c. Calculate b/a and enter score. N/A

CS.1(1) Have specific standards been established for design representations (e.g., HIPO charts, program design language, flow charts, data flow diagrams)? Y N N/A

CS.1(2) Have specific standards been established for calling sequence protocol between software units? Y N N/A

CS.1(3) Have specific standards been established for external I/O protocol and format for all software units? Y N N/A

CS.1(4) Have specific standards been established for error handling for all software units? Y N N/A

CS.1(5) Do all references to the same CSCI function use a single, unique name? Y N N/A

CS.2(1) Have specific standards been established for all data representation in the design? Y N N/A

CS.2(2) Have specific standards been established for the naming of all data? Y N N/A

CS.2(3) Have specific standards been established for the definition and use of all global variables? Y N N/A

CS.2(4) Are there procedures for establishing consistency and concurrency of multiple copies (e.g., copies at different nodes) of the same software or data base version? Y N N/A

CS.2(5) Are there procedures for verifying consistency and concurrency of multiple copies (e.g., copies at different nodes) of the same software or data base version? Y N N/A

CS.2(6) Do all references to the same data use a single, unique name? Y N N/A

DI.1(1) Is a graphic portrayal (e.g., figures, diagrams, tables) provided which identifies all software functions and functional interfaces? Y N N/A

DI.1(2) Is a graphic portrayal provided which identifies all different types of CSCI-level information and the information flow within the CSCI? Y N N/A

DI.1(3) Are there requirements for the organization and distribution of information within the CSCI (e.g., information distributed across nodes or among storage devices)? Y N N/A

DI.1(4) Are there requirements for file/library accessibility from each node? Y N N/A

DI.1(6) Are there requirements to distribute all mission-critical functions over redundant elements or nodes? Y N N/A

DI.1(7) Are there requirements to distribute control functions across different nodes/elements so as to ensure system operation under anomalous conditions? Y N N/A

DI.1(8) Do the requirements allow for implementing functions across several physical structures (i.e., function and physical structure are not necessarily the same)? Y N N/A

DI.1(9) Are there requirements regarding the number of nodes that can be removed from an operational system such that each remaining node still maintains its capability to communicate with all other remaining nodes? Y N N/A

DO.1(1)	Are current versions of all software documentation related to the project free from access control (i.e., any member of the current project or other projects may access a copy of any document)?	Y N N/A
DO.2(1)	Is all the documentation structured and written clearly and simply such that procedures, functions, algorithms, etc. can be easily understood?	Y N N/A
DO.2(2)	Does the requirements/design documentation clearly depict control and data flow (e.g., graphic portrayed with accompanying explanations or PDL)?	Y N N/A
DO.2(3)	Does all documentation contain an indexing scheme which facilitates quickly locating and accessing various information in the document (e.g., hierarchical structured table of contents, inserted tabs)?	Y N N/A
DO.2(4)	Do the software specifications and design documentation have separate volumes or separations within a single volume based on system functions or software functions?	Y N N/A
DO.2(5)	Does the documentation completely characterize the operational capabilities of the software (e.g., identify all the performance parameters and limitations?	Y N N/A
DO.2(6)	Does the documentation contain comprehensive descriptions of all system/software functions including functional processing, functional algorithms, and functional interfaces?	Y N N/A
EC.1(1)	Have performance requirements and limitations for system communication efficiency been specified for each CSCI function?	Y N N/A
EP.1(1)	Have performance requirements and limitations for processing efficiency been specified for each CSCI function (e.g., flow time for process, execution time)?	Y N N/A
EP.1(3)	Is there a requirement to use an optimizing compiler or to code in assembly language to optimize processing efficiency?	Y N N/A

EP.1(5)	Is memory management of the CSCI free from requirements for overlays?	Y N N/A
EP.2(1)	Have performance requirements and limitations for storing data to efficiently process it been specified for each CSCI function?	Y N N/A
EP.2(2)	Are there requirements to efficiently process stored information (e.g., rapidly update files, buffers, etc.)?	Y N N/A
EP.2(3)	Does the source code language(s) enable variable initialization when the variable is declared?	Y N N/A
ES.1(1)	Have performance requirements and limitations for storing data to efficiently utilize primary and secondary storage been specified for each CSCI function (e.g., data packing, dynamic memory management)?	Y N N/A
ES.1(2)	Does the memory management of the CSCI incorporate virtual storage?	Y N N/A
ES.1(5)	Does the memory management of the CSCI incorporate dynamic reallocation of physical memory space during execution (dynamic memory management)?	Y N N/A
ES.1(7)	Is there a requirement to use an optimizing compiler or to code in assembly language to optimize storage efficiency?	Y N N/A
ES.1(8)	Are there requirements to avoid redundant storage of files and libraries?	Y N N/A
FO.1(1)	a. How many functions in this CSCI?	____ N/A
	b. How many CSCI functions are duplicated in interoperating systems?	____ N/A
	c. Calculate 1-(b/a) and enter score.	____ N/A
FO.1(2)	a. How many duplicated function sets exist between this CSCI and interoperating CSCI's (i.e., the same function is performed in this CSCI and in an interoperating CSCI)?	____ N/A

b. In how many instances of the duplicated function sets will duplicated functions be deleted (i.e., leaving the responsibility for performing the function with one system)? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FO.1(3) a. How many duplicated function sets exist between this CSCI and interoperating CSCI's (i.e., the same function is performed in this CSCI and in an interoperating CSCI)? [ ] N/A

b. How many of the duplicated function sets require synchronization of the functions within the set? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FO.1(4) a. How many duplicated function sets exist between this CSCI and interoperating CSCI's (i.e., the same function is performed in this CSCI and in an interoperating CSCI)? [ ] N/A

b. How many of the duplicated function sets require redundancy management techniques/logic to enable system interoperability? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FS.2(1) Are there requirements to construct functions in such a way so as to facilitate their use in other similar CSCI applications? Y N N/A

FS.2(2) a. How many CSCI functions? [ ] N/A

b. How many CSCI functions are constructed in such a way so as to facilitate their use in other similar CSCI applications? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

FS.2(3) Are all inputs documented as to the specific use and limitations of the data? Y N N/A

FS.2(4) Are all input/output formats specified and documented? Y N N/A

FS.2(5) Are all outputs documented as to the specific use and interpretation of the data? Y N N/A

FS.2(6) a. How many CSCI functions? ____ N/A

b. How many CSCI functions will likely satisfy the requirements of other similar applications? ____ N/A

c. Calculate 1-(b/a) and enter score. ____ N/A

FS.3(1) Are there requirements to provide the user options for computation and output (e.g., selection of type of coordinate system, output media, format)? Y N N/A

FS.3(2) Are there requirements to enable modification of the resources allocated to functions (e.g., changing the amount of memory work space for a function)? Y N N/A

FS.3(3) a. How many functions are typically performed by a CSCI for this application? ____ N/A

b. How many functions does this CSCI perform? ____ N/A

c. Calculate b/a and enter score. (Note that if b/a is greater than one, enter one.) ____ N/A

ID.1(2) Is there a requirement to use a standard subset of the implementation language(s)? Y N N/A

ID.1(3) Has a standard subset of the implementation language(s) been established for coding? Y N N/A

ID.2(1) Are the same version and dialect of the implementation language(s) supported on other machines? Y N N/A

MO.1(1) Are there requirements to develop all software functions and software elements according to structured design techniques (e.g., top-down design)? Y | N | N/A

MO.1(2) Are all software functions and CSCI's developed according to structured design techniques? Y | N | N/A

MO.2(1) Are there requirements regarding the relationships among software entities (i.e., types of coupling allowed among software functions, CSCI's, CSC's and units) (e.g., requirements to minimize content, common and external coupling among software entities)? Y | N | N/A

MO.2(2) a. How many interfaces among software functions? ____ N/A

b. How many software functional interfaces include:

b1 Content coupling ____ N/A

b2 Common coupling ____ N/A

b3 External coupling ____ N/A

c. Calculate 1 - ((b1+b2+b3)/(3xa)) and enter score. ____ N/A

MO.2(3) a. How many interfaces among software functions? ____ N/A

b. How many software functional interfaces include:

b1 Control coupling ____ N/A

b2 Stamp coupling ____ N/A

b3 Data coupling ____ N/A

c. Calculate 1 - ((b1+b2)/(2xa)+b3/a) and enter score. ____ N/A

MO.2(4) Are there requirements regarding the relationship between the elements within a software entity (i.e., cohesion value for software functions, CSCI's, CSC's, and units) (e.g., all software entities are required to reflect an average cohesion value of 0.6 or greater)? Y N N/A

MO.2(5) What is the average cohesion value of all software functions in this CSCI?
List each software function and its cohesion value below: ___ N/A

SOFTWARE FUNCTION	COHESION VALUE

OP.1(1) Have the operating characteristics of the CSCI been specified (i.e., the normal and alternate procedures and actions performed by the CSCI? Y N N/A

OP.1(2) Are all the errors specified which are to be reported to the operator/user? Y N N/A

OP.1(3) Are the required operator/user responses specified for all reported errors? Y N N/A

OP.1(4) Are there requirements to provide the operator with the capability to interrupt system operation, obtain operational status, save and enter data, and continue processing? Y N N/A

OP.1(5) a. How many operations/responses are performed by the operator for a typical mission/job? ___ N/A

b. Calculate $1/(1+a)$ and enter score. ___ N/A

1-17

OP.1(6) Are there requirements to specify the procedures for setting up a mission/job and completing it? Y N N/A

OP.1(8) Are there requirements to maintain a hard copy log of all operator interactions with the CSCI? Y N N/A

OP.1(10) Are there requirements to provide simple and consistent operator messages and require simple and consistent operator responses (i.e., minimize the number of operator message and response formats; use the same format types throughout the CSCI)? Y N N/A

OP.1(11) Are there requirements to report all access violations to the operator? Y N N/A

OP.1(12) Are there requirements specifying the appropriate response(s) (by the operator, the system/software, or both) for all access violations? Y N N/A

OP.1(13) Are there requirements to provide the operator/software the capability to obtain specific system (or network) resource status information and to reallocate resources? Y N N/A

OP.1(14) Are there requirements to provide the operator/user the capability to select different nodes for different types of processing or for retrieval of information? Y N N/A

OP.1(15) Are there requirements to provide the operator/user the capability to manipulate data regardless of the data's location in the system? Y N N/A

OP.1(16) Are there requirements to make system implementation details transparent to the user (e.g., the user can access a file without knowing its location in the system/network)? Y N N/A

OP.2(1) a. What is the total number of different user input parameters? ____ N/A

b. How many input parameters have default values? ____ N/A

	c.	Calculate b/a and enter score.	[ ] N/A
OP.2(2)	a.	How many different input formats must the user be familiar with?	[ ] N/A
	b.	Calculate 1/a and enter score.	[ ] N/A
OP.2(3)	a.	What is the total number of different user input parameters?	[ ] N/A
	b.	How many user input parameters enable the user to provide a description along with the values (e.g., user inputs: "targets = 2")?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
OP.2(4)		Are there requirements to enable the user to review and modify all input data prior to execution?	Y N N/A
OP.2(5)		Are there requirements to terminate all user-input data by explicitly defined logical end of input?	Y N N/A
OP.2(6)		Are there requirements to provide the user options for input media (e.g., terminal, tape drive, card reader)?	Y N N/A
OP.3(1)		Are there requirements to provide the user with output control (e.g., choosing specific outputs, output media, output formats, amount of output)?	Y N N/A
OP.3(2)		Is there a requirement for all outputs to the user to have unique, descriptive labels for identifying data?	Y N N/A
OP.3(3)		Is there a requirement to provide all outputs to the user in user-oriented measurement units?	Y N N/A

OP.3(4)	a. How many different formats are output to the user (e.g., CRT display arrangements, printer outputs)?	___ N/A
	b. Calculate 1/a and enter score.	___ N/A
OP.3(5)	Are all user outputs separated into logical groups to facilitate user examination?	Y N N/A
OP.3(6)	Are there requirements for all error messages to clearly identify the nature of the error to the user?	Y N N/A
OP.3(7)	Are there requirements to provide the user with options for output media?	Y N N/A
OP.3(8)	Are there requirements to establish a standard (common) user command language for network information and data access?	Y N N/A
RE.1(1)	Are there requirements to ensure communication paths to all remaining nodes/communication links in the event of a failure of one node/link?	Y N N/A
RE.1(2)	Are there requirements for maintaining the integrity of all data values following the occurrence of anomalous conditions?	Y N N/A
RE.1(3)	Are there requirements to enable all disconnected nodes to rejoin the network after recovery, such that the processing functions of the system are not interrupted?	Y N N/A
RE.1(4)	Are there requirements to replicate all critical data in the CSCI at two or more distinct nodes?	Y N N/A
SD.2(1)	Has the specific standard been established that each unit prologue contain tne unit's function, author, version number, version date, inputs, outputs, assumptions and limitations.	Y N N/A

SD.2(2)	Has a standard been established for the identification and placement of comments in the unit?	Y N N/A
SD.3(5)	Has a standard format for the structure of units been established?	Y N N/A
SI.1(1)	Are there diagrams identifying all functions in a structured fashion (e.g., top-down hierarchical)?	Y N N/A
SI.1(8)	Are there requirements for a programming standard?	Y N N/A
SI.1(9)	Has a programming standard been established?	Y N N/A
SI.2(1)	Are there requirements to use a structured language or preprocessor to implement the software?	Y N N/A
SI.4(13)	Are there requirements for a programming standard?	Y N N/A
SS.1(1)	Are there requirements to control user input/output access in the CSCI (e.g., user access is limited by identification and password checking)?	Y N N/A
SS.1(2)	Are there requirements to control data access in the CSCI?	Y N N/A
SS.1(3)	Are there requirements to control the scope of task operations during execution (e.g., tasks cannot invoke other tasks, access system registers, or use privileged commands)?	Y N N/A
SS.1(4)	Are there requirements to control access to the network?	Y N N/A
SS.2(1)	Are there requirements to record and report all access to the system (e.g., record terminal and processor linkage, data file access, and jobs run information)?	Y N N/A
SS.2(2)	Are there requirements to immediately indicate and identify all access violations?	Y N N/A

ST.3(1)	Are there requirements to isolate I/O functions from computational functions?	Y N N/A
ST.3(2)	Are I/O functions isolated from computation functions?	Y N N/A
SY.1(1)	Are there requirements for the I/O transmission rates of this CSCI to be the same as the interoperating CSCI(s)?	Y N N/A
SY.1(2)	Are there requirements for this CSCI to use the same communication protocol as the interoperating CSCI(s)?	Y N N/A
SY.1(3)	Are there requirements for common interpretation of the content in all messages sent from and received by this CSCI and by the interoperating system(s) (e.g., all variables in the message have the same meaning)?	Y N N/A
SY.1(4)	Are there requirements for this CSCI to use the same structure and sequence for message contents as the interoperating system(s) (e.g., all real variables are 16 bits in length; all real coordinates are ordered XCOORD, YCOORD, ZCOORD)?	Y N N/A
SY.2(1)	Are there requirements for this CSCI to use the same data format as the interoperating system(s) (e.g., all characters are represented in ASCII format)?	Y N N/A
SY.2(2)	Are there requirements for this CSCI to establish the same data base structure as the interoperating CSCI(s) (e.g., all CSCI's use a relational data base containing similar information)?	Y N N/A
SY.2(3)	Are there requirements for this CSCI to provide the same data base access techniques as the interoperating CSCI(s)?	Y N N/A
SY.3(1)	Are there requirements for this CSCI to use the same word length as the interoperating CSCI(s)?	Y N N/A
SY.3(2)	Does this CSCI use the same word length as the interoperating CSCI(s)?	Y N N/A

SY.3(3) Are there requirements for this CSCI to use the same interrupt structure as the interoperating CSCI(s)? Y N N/A

SY.3(4) Does this CSCI use the same interrupt structure as the interoperating CSCI(s)? Y N N/A

SY.3(5) Are there requirements for this CSCI to use the same instruction set as the interoperating CSCI(s)? Y N N/A

SY.3(6) Does this CSCI use the same instruction set as the interoperating CSCI(s)? Y N N/A

SY.4(1) Are there requirements for this CSCI to use the same source code language(s) as the interoperating CSCI(s)? Y N N/A

SY.4(2) Are there requirements for this CSCI to use the same operating system as the interoperating CSCI(s)? Y N N/A

SY.4(3) Are there requirements for this CSCI to use the same support software as the interoperating CSCI(s)? Y N N/A

SY.5(1) Is documentation available from the interoperating system(s) that enables interoperability requirements to be established for this CSCI (e.g., documentation is up-to-date, complete, and clearly organized)? Y N N/A

TC.1(1) Is there a table(s) tracing all of the CSCI's allocated requirements to the parent system or subsystem specification(s)? Y N N/A

TN.1(1) Are there requirements to provide lesson plans and training materials for operators, and users, and maintainers of the CSCI? Y N N/A

TN.1(2) Are there requirements to provide realistic simulation exercises for the CSCI? Y N N/A

TN.1(3) Are there requirements to provide "help" information and diagnostic information for the operator, end user, and maintainer of the CSCI (e.g., provide an on-line list of legal commands or a list of the sequential steps in a process)? Y N N/A

TN.1(4) Are there requirements to provide selectable levels of aid and guidance for CSCI users of different degrees of expertise? Y N N/A

VR.1(1) Are there requirements to make system implementation details transparent to the user (e.g., the user can create a file without specifying its location in the system/network)? Y N N/A

GLOSSARY

Anomalous Condition: An event resulting in a deviation from the normal operating environment or procedures.

Cohesion Value: The type of relationship that exists among the elements of each software entity (Function, CSCI, Unit). The following are relative values for seven types of cohesion.

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohension.

1) Coincidental
   - No meaningful relationships among the elements of an entity.
   - Difficult to describe the module's function(s).
2) Logical
   - Entity performs (at each invocation) one of a class of related functions (e.g., "edit all data").
   - Entity performs more than one function.
3) Classical
   - Entity performs one of a class of functions that are related in time (Program procedure).
   - Entity performs more than one function.
4) Procedural
   - Entity performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).

GLOSSARY (continued)

5) Communicational
- Entity has procedural strength; in addition, all of the elements "communicate" with one another (e.g., reference same data or pass data among themselves).
- All functions use the same data.

6) Informational
- Entity performs multiple functions where the functions (entry points in the module) deal with a single data structure.
- Physical packaging together of two or more entities having functional strength.
- All functions use the same data.

7) Functional
- All entity elements are related to the performance of a single function.

Command Language: The set of instructions used to invoke specific operations in a computer software subsystem/program.

Communication Channel: The pathways along which data/messages are communicated to the various system components or nodes (i.e., other computer, data storage units, special processors, etc.).

Coupling: The type of relationship that exists between two software entities (Functions, CSCIs, Units). In achieving a highly modular design it is essential to minimize the relationships among software entities. The goal is to design software entities with low coupling. The scae of coupling from worst to best is: (1) Content Coupling, 2) Common Coupling, 3) External Coupling, 4) Control Coupling, 5) Stamp Coupling, and 6) Data Coupling.

1) Content Coupling - One software entity references the contents of another software entity.

2) Common Coupling - Software entities reference a shared global data structure.

GLOSSARY (continued)

3) External Coupling - Software entities reference the same externally declared symbol.

4) Control Coupling - One software entity passes control elements as arguments to another software entity.

5) Stamp Coupling - Two software entities reference the same data structure, which is not global.

6) Data Coupling - One software entity calls another and the software entities are not coupled as defined above (in 1 thorugh 5).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Format: The positioning, packing or organization of the order that the data appears in.

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Record: A structured grouping of related data elements for the purpose of storage or transmission.

Data Reference: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Database Management Scheme: The methods and commands used to access or operate the database management software system.

Design Representation: A formal statement of the details or organization of a design using one of a number of design representation methodologies, such as, Flow Charts, HIPO Charts, PDL, etc.

I/O Channel: The pathways along which data/messages are communicated to the various user-oriented peripherals in the system (e.g., CRT, Printer).

Microcode Instruction: A low-level computer instruction specifying a single machine operation.

GLOSSARY (continued)

Mission-Critical Function: A feature essential to fulfilling the desired objectives of the system.

Multiple Transfer Index Parameters: A value used to select a variation in the order of code execution (i.e., case statement, program switch, etc.).

Network: A system of computers, terminals, and data bases that are linked/ interconnected with the use of communication lines.

Node: The points at which subsidiary parts originate or connect to a system containing interconnected system parts or devices.

Range-Test: A test performed to validate the object of interest over the complete spectrum of applicable values.

Subscript Value: A value used to reference an entity from a group of related objects (i.e., table index, array index, etc.).

Synchronization: The process of ensuring that two or more components of a system are ready and capable of communicating with one another.

Virtual Storage: The storage space that may be regarded as addressible main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

WORKSHEET PREPARATION INSTRUCTIONS

SECTION A - GENERAL INFORMATION

1. Enter the name and contract number of the project.
2. Enter the date the worksheet is prepared (month, day, year).
3. Enter the name of the person responsible for completing the worksheet.
4. Enter the name of the entity to which the worksheet is applied.
5. Enter the number and name of all documentation which is used as source material for answering the metric questions in Section B.
6. Enter comments reflecting the inspector's observations on product quality and any additional information regarding specific metric questions. Attach additional sheets, as necessary.

SECTION B - METRIC QUESTIONS

Answer all applicable metric questions by circling the appropriate answer (Y = yes, N = no, N/A = not applicable) or by entering the appropriate value. A glossary of terms is provided at the end of this section.

METRIC WORKSHEET 2 CSCI LEVEL

SECTION A - GENERAL INFORMATION

1. PROJECT ______________________
2. DATE ______________________
3. INSPECTOR ______________________
4. PRODUCT ______________________
5. SOURCE DOCUMENTATION:

____
____
____
____
____
____
____

6. INSPECTOR COMMENTS:

2-2

SECTION B - METRIC QUESTIONS

AC.1(7)	Do the numerical techniques used in implementing applicable functions (e.g., mission-critical functions) provide enough precision to support accuracy objectives?	Y N N/A
AM.3(1)	Are there provisions for recovery from all computational failures?	Y N N/A
AM.4(1)	Are there provisions for recovery from all detected hardware faults (e.g., arithmetic faults, hardware failure, clock interrupt)?	Y N N/A
AM.5(1)	Are there provisions for recovery from all I/O device errors?	Y N N/A
AM.6(1)	Are there provisions for recovery from all communication transmission errors?	Y N N/A
AM.6(2)	Is error checking information (e.g., checksum, parity bit) computed and transmitted with all messages?	Y N N/A
AM.6(3)	Is error checking information computed and compared with all message receptions?	Y N N/A
AM.6(4)	Are transmission retries limited for all transmissions?	Y N N/A
AM.7(1)	Are there provisions for recovery from all failures to communicate with other nodes or other systems?	Y N N/A
AM.7(2)	Are there provisions to periodically check all adjacent nodes or interoperating systems for operational status?	Y N N/A
AM.7(3)	Are there provisions for alternate routing of messages?	Y N N/A
AP.1(1)	Are there provisions for limiting specific references to the data base management scheme (e.g., all data calls for data base information are processed through an executive)?	Y N N/A

AP.5(1)	a.	How many algorithms?	N/A
	b.	How many algorithms are not unique to this application (e.g., table driven algorithm)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.5(2)	a.	How many algorithms?	N/A
	b.	How many algorithms have been verified with respect to their requirements?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.5(3)	a.	How many algorithms?	N/A
	b.	How many algorithms have test data available which reflects results of algorithm verification?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.1(2)	a.	What is the total memory space allocated?	N/A
	b.	What is the estimated memory space used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
AT.1(3)	a.	What is the total auxiliary storage space allocated?	N/A
	b.	What is the estimated auxiliary storage space used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A

AT.2(3) a. What is the total processing time allocated? [ ] N/A

b. What is the estimated I/O processing time used (total less spare)? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

AT.3(1) a. What is the total I/O channel time allocated? [ ] N/A

b. What is the estimated I/O channel time used (total less spare)? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

Note, for more than one I/O channel, list answers to a., b., and c. below, and enter average of answers in boxes above.

I/O CHANNEL	a.	b.	c.
______	___	___	___
______	___	___	___
______	___	___	___
______	___	___	___

AT.3(2) a. What is the total communication channel time allocated? [ ] N/A

b. What is the estimated communication channel time used (total less spare)? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

Note. for more than one communication channel, list answers to a., b., and c. below and enter average of answers in boxes above.

COMMUNICATION CHANNEL	a.	b.	c.
____	____	____	____
____	____	____	____
____	____	____	____
____	____	____	____

AT.4(1) Are all processors, communication links, memory devices, and peripherals compatible (e.g., of a common vendor or model)? Y N N/A

AU.1(1) Are all processes and functions partitioned to be logically complete and self-contained so as to minimize interface complexity? Y N N/A

AU.1(4) a. How much estimated processing time is typically spent executing the entire CSCI? [ ] N/A

b. How much estimated processing time is typically spent in execution of hardware and device interface protocol? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

AU.2(2) Does the executive software perform testing of its own operation and of the communication links, memory devices, and peripheral devices? Y N N/A

2-6

CL.1(2)	Is there compliance with the network communication protocol standards?		Y N N/A
CL.1(3)	Is all network processing controlled in accordance with the protocol standard?		Y N N/A
CL.1(4)	Are all user sessions controlled in accordance with the protocol standard?		Y N N/A
CL.1(5)	Is all communication routing performed in accordance with the protocol standard?		Y N N/A
CL.1(6)	Is all message handling (e.g., synchronization, message decoding) performed in a uniform manner in accordance with the protocol standard?		Y N N/A
CL.1(7)	a.	How many top-level CSC's receive inputs from other systems?	___ N/A
	b.	Calculate 1/a and enter score.	___ N/A
CL.1(8)	a.	How many top-level CSC's transmit outputs to other systems?	___ N/A
	b.	Calculate 1/a and enter score.	___ N/A
CL.1(11)	Are there no time-critical constraints with respect to external communication (e.g., data freshness)?		Y N N/A
CL.1(13)	Is there a general description of how the computer system appears to the users and how the users interact with the computer system (e.g., operational concept document)?		Y N N/A
CL.1(14)	Is there a complete and definitive set of operating procedures for using this system?		Y N N/A
CL.2(1)	Do all data representations and translations comply with the established standard?		Y N N/A
CL.2(2)	a.	How many top-level CSC's perform data translations?	___ N/A
	b.	Calculate 1/a and enter score.	___ N/A

CL.2(4)	a.	How many different formats are used for input data from other systems (e.g., formats for data positioning, data packing, block transmission)?	☐ N/A
	b.	Calculate 1/(1+a) and enter score.	☐ N/A
CL.2(6)	a.	How many different formats are used for output data to other systems (e.g., formats for data positioning, data packing, block transmission)?	☐ N/A
	b.	Calculate 1/(1+a) and enter score.	☐ N/A
CP.1(1)		Are all inputs, processing, and outputs clearly and precisely defined?	Y N N/A
CP.1(2)	a.	How many data references are identified?	☐ N/A
	b.	How many identified data references are documented with regard to source, meaning, and format?	☐ N/A
	c.	Calculate b/a and enter score.	☐ N/A
CP.1(3)	a.	How many data items are defined (i.e., documented with regard to source, meaning, and format)?	☐ N/A
	b.	How many defined data items are referenced?	☐ N/A
	c.	Calculate b/a and enter score.	☐ N/A
CP.1(4)	a.	How many data references are identified?	☐ N/A
	b.	How many identified data references are computed or obtained from an external source (e.g., referencing global data with preassigned values, input parameters with preassigned values)?	☐ N/A
	c.	Calculate b/a and enter score.	☐ N/A

CP.1(6)	Have all functions for this CSCI been allocated to top-level CSC's of this CSCI?	Y N N/A
CP.1(9)	Are all conditions and alternative processing options defined for each decision point?	Y N N/A
CP.1(11)	a. How many software problem reports have been recorded, to date?	N/A
	b. How many recorded software problem reports have been closed (resolved), to date?	N/A
	c. Calculate b/a and enter score.	N/A
CS.1(1)	Are the design representations in the formats of the established standard?	Y N N/A
CS.1(5)	Do all references to the same top-level CSC use a single, unique name?	Y N N/A
CS.2(1)	Does all data representation comply with the established standard?	Y N N/A
CS.2(2)	Does the naming of all data comply with the established standard?	Y N N/A
CS.2(3)	Is the definition and use of all global variables in accordance with the established standard?	Y N N/A
CS.2(4)	Are there procedures for establishing consistency and concurrency of multiple copies (e.g., copies at different nodes) of the same software or data base version?	Y N N/A
CS.2(5)	Are there procedures for verifying consistency and concurrency of multiple copies (e.g., copies at different nodes) of the same software or data base version?	Y N N/A
CS.2(6)	Do all references to the same data use a single, unique name?	Y N N/A

DI.1(1)	Is a graphic portrayal (e.g., figures, diagrams, tables) provided which identifies all top-level CSC's and their interfaces within the CSCI?	Y N N/A
DI.1(2)	Is a graphic portrayal provided which identifies all types of top-level CSC information groupings and top-level CSC information flow within the CSCI?	Y N N/A
DI.1(4)	Are all files/libraries accessible from each node in accordance with the established requirements?	Y N N/A
DI.1(5)	Can alternate processing sources (for this CSCI) be selected within the system?	Y N N/A
DI.1(6)	Are all mission-critical functions (for this CSCI) distributed over redundant elements or nodes?	Y N N/A
DI.1(7)	Are control functions (for this CSCI) distributed across different nodes/elements so as to ensure system operation under anomalous conditions?	Y N N/A
DI.1(9)	Can each node communicate with all remaining nodes in accordance with the requirements concerning node removal?	Y N N/A
DO.1(1)	Are current versions of all software documentation related to the project free from access control (i.e., any member of the current project or other projects may access a copy of any document)?	Y N N/A
DO.2(1)	Is all the documentation structured and written clearly and simply such that procedures, functions, algorithms, etc. can be easily understood?	Y N N/A
DO.2(2)	Does the requirements/design documentation clearly depict control and data flow (e.g., graphic portrayed with accompanying explanations or PDL)?	Y N N/A
DO.2(3)	Does all documentation contain an indexing scheme which facilitates quickly locating and accessing various information in the document (e.g., hierarchical structured table of contents, inserted tabs)?	Y N N/A

DO.2(4)	Do the software specifications and design documentation have separate volumes or separations within a single volume based on system functions or software functions?	Y N N/A
DO.2(5)	Does the documentation completely characterize the operational capabilities of the software (e.g., identify all the performance parameters and limitations)?	Y N N/A
DO.2(7)	Does the documentation contain comprehensive descriptions of all algorithms used and their limitations, including inputs, outputs, and required precision?	Y N N/A
EP.1(5)	a. How many different overlays are used in this CSCI?	___ N/A
	b. Calculate 1/a and enter score.	___ N/A
EP.2(2)	Has the storage of all information (e.g., files, code, arrays, buffers) been organized for efficient processing (e.g., minimum search time)?	Y N N/A
EP.2(3)	Does the source code language(s) enable variable initialization when the variable is declared?	Y N N/A
EP.2(6)	Does the method(s) for relating similar data items facilitate efficient processing (e.g., arrays, doubly linked lists, directories)?	Y N N/A
ES.1(2)	Does the memory management of the CSCI incorporate virtual storage?	Y N N/A
ES.1(5)	Does the memory management of the CSCI incorporate dynamic reallocation of physical memory space during execution (dynamic memory management)?	Y N N/A
ES.1(8)	Is the CSCI free from redundant storage of files and libraries (e.g., duplicate copies are not stored at different nodes in a network; multiple versions of the same file are not part of the working CSCI)?	Y N N/A
FS.2(2)	Does the design implementation the CSCI functions in such a way so as to facilitate their use in other similar CSCI applications?	Y N N/A

FS.2(3)	Are all inputs documented as to the specific use and limitations of the data?	Y N N/A
FS.2(4)	Are all input/output formats specified and documented?	Y N N/A
FS.2(5)	Are all outputs documented as to the specific use and interpretation of the data?	Y N N/A
FS.3(1)	Can the user choose among computation and output options (e.g., user selecting type of coordinate system, output media, format)?	Y N N/A
FS.3(2)	Can the resources allocated to functions be modified (during execution)?	Y N N/A
ID.1(3)	Is the software free from using any non-standard constructs of the implementation language(s)?	Y N N/A
ID.2(1)	Are the same version and dialect of the implementation language(s) supported on other machines?	Y N N/A
MO.1(2)	Are all top-level CSC's developed according to structured design techniques?	Y N N/A
MO.1(9)	Does each top-level CSC's have a single processing objective (i.e., all the processes within the top-level CSC are related to the same objective)?	Y N N/A
MO.2(2)	a. How many interfaces among top-level CSC's?	____ N/A
	b. How many top-level CSC interfaces include:	
	b1. Content coupling	____ N/A
	b2. Common coupling	____ N/A

METRIC WORKSHEET 2 CSCI LEVEL

b3. External coupling N/A

c. Calculate 1-((b1+b2+b3)/(3xa)) and enter score. N/A

MO.2(3) a. How many interfaces among top-level CSC's? N/A

b. How many top-level CSC interfaces include:

b1. Control coupling N/A

b2. Stamp coupling N/A

b3. Data coupling N/A

c. Calculate ((b1+b2)/(2xa)+b3/a) and enter score. N/A

MO.2(5) What is the average cohesion value of all top-level CSC's in this CSCI? N/A

List each top-level CSC and its cohesion value below.

TOP-LEVEL CSC	COHESION VALUE

OP.1(1)	Has a description of the operating characteristics of the CSCI been provided (i.e., the normal and alternate procedures and actions performed by the CSCI) (e.g., operating characteristics are described in an operator's manual)?	Y N N/A
OP.1(2)	Are all the errors reported to the operator/user as specified in the requirements?	Y N N/A
OP.1(3)	Is the capability provided for operator/user response to all reported errors as specified in the requirements?	Y N N/A
OP.1(4)	Can the operator interrupt system operation, save and enter data, and continue processing?	Y N N/A
OP.1(7)	Are the procedures specified for setting up a mission/job and completing it?	Y N N/A
OP.1(9)	Is a hard copy log provided of all operator interactions with the system/CSCI?	Y N N/A
OP.1(10)	Are all operator messages and responses simple and consistent (e.g., "Disk drive #2 is off-line"; "Enter 'YES' for retry or 'NO' to stop processing")?	Y N N/A
OP.1(11)	Are all access violations reported to the operator in accordance with the requirements?	Y N N/A
OP.1(12)	Are appropriate responses performed/provided for all access violations in accordance with the requirements?	Y N N/A
OP.1(13)	Can the operator/software obtain specific system (or network) resource status information and reallocate resources?	Y N N/A
OP.1(14)	Can the operator select different nodes for different types of processing or for retrieval of information?	Y N N/A

OP.1(15)	Can the operator/user manipulate data regardless of the data's location in the system?	Y | N | N/A
OP.1(16)	Are system implementation details transparent to the user (e.g., the user can access a file without knowing its location in the system/network)?	Y | N | N/A
OP.2(4)	Can the user review and modify all input data prior to execution?	Y | N | N/A
OP.2(5)	Are all user-input data terminated by explicitly defined logical end of input?	Y | N | N/A
OP.2(6)	Can the user select among options for input media (e.g., terminal, tape drive, card reader)?	Y | N | N/A
OP.3(1)	Can the user control output (e.g., choose specific outputs, output media, output formats, amount of output)?	Y | N | N/A
OP.3(2)	Do all outputs to the user have unique, descriptive labels for identifying data?	Y | N | N/A
OP.3(3)	Are all outputs to the user provided with user-oriented measurement units?	Y | N | N/A
OP.3(6)	Do all error messages clearly identify the nature of the error to the user?	Y | N | N/A
OP.3(7)	Can the user select among options for output media?	Y | N | N/A
OP.3(8)	Is there a standard (common) command language for network information and data access?	Y | N | N/A
RE.1(1)	Do communication paths exist to all remaining nodes/links in the event of a failure of one node/link?	Y | N | N/A

RE.1(2)	Is the integrity of all data values maintained following the occurrence of anomalous conditions?	Y N N/A
RE.1(3)	Can all disconnected nodes rejoin the network after recovery, such that the processing functions of the system are not interrupted?	Y N N/A
RE.1(4)	Are all critical data in the system (or CSCI) replicated at two or more distinct nodes, in accordance with specified requirements?	Y N N/A
SI.1(1)	Does the design of the CSCI reflect a structured design approach (e.g., top-down design)?	Y N N/A
SI.1(6)	a. How many unique data items are in common blocks?	____ N/A
	b. Calculate 1/a and enter score.	____ N/A
SI.1(7)	a. How many unique data items are in common blocks?	____ N/A
	b. How many unique common blocks?	____ N/A
	c. Calculate b/a and enter score.	____ N/A
SI.1(9)	Has a programming standard been established?	Y N N/A
SI.1(10)	Do the descriptions of all top-level CSC's identify all interfacing top-level CSC's and all interfacing hardware?	Y N N/A
SI.2(1)	Are there requirements to use a structured language or preprocessor to implement the software?	Y N N/A
SI.4(13)	Are there requirements for a programming standard?	Y N N/A
SS.1(1)	Are there controls on user input/output access in accordance with the specified requirements (e.g., user access is limited by identification and password checking)?	Y N N/A

SS.1(2) Are there controls on data access in the system (CSCI) in accordance with the specified requirements (e.g., authorization tables and privacy locks)? Y N N/A

SS.1(3) Are there controls on the scope of task operations during execution in accordance with the specified requirements (e.g., invoke other tasks, access system registers, or use privileged commands)? Y N N/A

SS.1(4) Are there controls on access to the network in accordance with the specified requirements? Y N N/A

SS.2(1) Is all access to the system recorded and reported in accordance with the specified requirements (e.g., terminal and processor linkage, data file access, and jobs run information)? Y N N/A

SS.2(2) Are all access violations immediately indicated and identified in accordance with the specified requirements? Y N N/A

ST.3(2) Is I/O isolated from computation in the design? Y N N/A

ST.3(5) Does each top-level CSC perform unique operations (i.e., similar operations are not performed within different top-level CSC's which could be restricted to single top-level CSC)? Y N N/A

ST.3(6) a. In how many top-level CSC's are non-related functions performed (i.e., functions which do not contribute to the same overall objective)? ____ N/A

b. Calculate 1/(1+a) and enter score. ____ N/A

List the number of non-related functions performed by each top-level CSC.

TOP-LEVEL CSC	NUMBER OF NON-RELATED FUNCTIONS
______	______
______	______
______	______
______	______

SY.1(1) Does this system (CSCI) use the same I/O transmission rate as the interoperating system(s) in accordance with the specified requirements? Y N N/A

SY.1(2) Does this system (CSCI) use the same communication protocol as the interoperating system in accordance with the specified requirements? Y N N/A

SY.1(3) Is there a common interoperation of the content in all messages sent from and received by this system (CSCI) and by the interoperating system(s) (CSCI) in accordance with the specified requirements (e.g., all variables in the message have the same meaning)? Y N N/A

SY.1(4) Does this system (CSCI) use the same structure and sequence for message contents as the interoperating system(s) (CSCI) in accordance with the specified requirements (e.g., all real variables are 16 bits in length; all real coordinates are ordered XCOORD, YCOORD, ZCOORD)? Y N N/A

SY.2(1) Does this system (CSCI) use the same data format as the interoperating system(s), in accordance with the specified requirements (e.g., all characters are represented in ASCII format)? Y N N/A

SY.2(2) Does this system (CSCI) use the same data base structure as the interoperating system(s), in accordance with the specified requirements (e.g., all systems use a relational data base containing similar information)? Y N N/A

SY.2(3) Does this system (CSCI) provide the same data base access techniques as the interoperating system(s), in accordance with the specified requirements? Y N N/A

SY.4(1)	Does this system (CSCI) use the same source code language(s) as the interoperating system(s), in accordance with the specified requirements?	Y N N/A
SY.4(2)	Does this system use the same operating system as the interoperating system(s), in accordance with the specified requirements?	Y N N/A
SY.4(3)	Does this system use the same support software as the interoperating system(s), in accordance with the specified requirements?	Y N N/A
TC.1(1)	Is there a table(s) tracing all of the top-level CSC allocated requirements to the parent CSCI specification?	Y N N/A
VR.1(1)	Are the system implementation details transparent to the user (e.g., the user can create a file without specifying its location in the system/network)?	Y N N/A

GLOSSARY

Anomalous Condition: An event resulting in a deviation from the normal operating environment or procedures.

Cohesion Value: The type of relationship that exists among the elements of each software entity (Function, CSCI, Unit). The following are relative values for seven types of cohesion:

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohension.

1) Coincidental
   - No meaningful relationahips among the elements of an entity.
   - Difficult to describe the module's function(s).
2) Logical
   - Entity performs (at each invocation) one of a class of related functions (e.g., "edit all data").
   - Entity performs more than one function.
3) Classical
   - Entity performs one of a class of functions that are related in time (Program procedure).
   - Entity performs more than one function.
4) Procedural
   - Entity performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).
5) Communicational

GLOSSARY (Co. tinued)

- Entity has procedural strength; in addit on, all of the elements "communicate" with one other (e.g., reference same data oı pass data among themselves).
- All functions use the same data.

6) Informational
   - Entity performs multiple functions where the functions (entry points in the module) deal with a single data structure.
   - Physical packaging together of two or more entities having functional strength.
   - All functions use the same data.

7) Functional
   - All entity elements are related to the performance of a single function.

Command Language: The set of instructions used to invoke specific operations in a computer software subsystem/program.

Communication Channel: The pathways along which data/messages are communicated to the various system components or nodes (i.e., other computer, data storage units, special processors, etc.).

Coupling: The type of relationship that exists between two so. tware entities (Functions, CSCIs, Units). In achieving a highly modular design it is essential to minimize the relationships among software entities. The goal is to design software entities with low coupling. The scae of coupling from worst to best is: (1) Content Coupling, 2) Common Coupling, 3) External Coupling, 4) Control Coupling, 5) Stamp Coupling, and 6) Data Coupling.

1) Content Coupling - One software entity references the contents of another software entity.

2) Common Coupling - Software entities reference a shared global data structure.

3) External Coupling - Software entities reference the same externally declared symbol.

4) Control Coupling - One software entity passes control elements as arguments to another software entity.

GLOSSARY (Continued)

5) Stamp Coupling - Two software entities reference the same data structure, which is not global.

6) Data Coupling - One software entity calls another and the software entities are not coupled as defined above (in 1 through 5).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Reference: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Database Management Scheme: The methods and commands used to access or operate the database management software system.

Design Representation: A formal statement of the details or organization of a design using one of a number of design representation methodologies, such as, Flow Charts, HIPO Charts, PDL, etc.

I/O Channel: The pathways along which data/messages are communicated to the various user-oriented peripherals in the system (e.g., CRT, Printer).

Mission-Critical Function: A feature essential to fulfilling the desired objectives of the system.

Network: A system of computers, terminals, and data bases that are linked/ interconnected with the use of communication lines.

Node: The points at which subsidiary parts originate or connect to a system containing interconnected system parts or devices.

GLOSSARY (Continued)

Virtual Storage: The storage space that may be regarded as addressible main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

WORKSHEET PREPARATION INSTRUCTIONS

SECTION A - GENERAL INFORMATION

1. Enter the name and contract number of the project.
2. Enter the date the worksheet is prepared (month, day, year).
3. Enter the name of the person responsible for completing the worksheet.
4. Enter the name of the entity to which the worksheet is applied.
5. Enter the number and name of all documentation which is used as source material for answering the metric questions in Section B.
6. Enter comments reflecting the inspector's observations on product quality and any additional information regarding specific metric questions. Attach additional sheets, as necessary.

SECTION B - METRIC QUESTIONS

Answer all applicable metric questions by circling the appropriate answer (Y = yes, N = no, N/A = not applicable) or by entering the appropriate value. A glossary of terms is provided at the end of this section.

SECTION A - GENERAL INFORMATION

1. PROJECT ______________________
2. DATE ______________________
3. INSPECTOR ______________________
4. PRODUCT ______________________
5. SOURCE DOCUMENTATION:

____
____
____
____
____
____
____

6. INSPECTOR COMMENTS:

METRIC WORKSHEET 3A CSCI LEVEL

SECTION B - METRIC QUESTIONS

AM.1(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AM.2(2)		Are values of all applicable external inputs with range specifications checked with respect to specified range prior to use?	Y N N/A
AM.2(3)		Are all applicable external inputs checked with respect to specified conflicting requests prior to use?	Y N N/A
AM.2(4)		Are all applicable external inputs checked with respect to specified illegal combinations prior to use?	Y N N/A
AM.2(5)		Are all applicable external inputs checked for reasonableness before processing begins?	Y N N/A
AM.2(6)		Are all detected errors, with respect to applicable external inputs, reported before processing begins?	Y N N/A
AM.2(7)	a.	How many applicable units (answers of Y or N on 3B)	N/A
	b.	How many units with answer of Y (see 3B)	N/A
	c.	Calculate b/a and enter score.	N/A
AM.3(2)		Are all critical loop and multiple transfer index parameters (e.g., supporting a mission-critical function) checked for out-of-range values before use?	Y N N/A
AM.3(3)		Are all critical subscripts (e.g., supporting a mission-critical function) checked for out-of-range values before use?	Y N N/A

AM.3(4)		Are all critical output data (e.g., supporting a mission-critical function) checked for reasonable values prior to final outputting?	Y N N/A
AP.1(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(1)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(2)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(4)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 3A | CSCI LEVEL

AP.3(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.4(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.1(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.1(2)	a.	What is the total memory space allocated?	N/A
	b.	What is the estimated memory space used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
AT.1(3)	a.	What is the total auxiliary storage space allocated?	N/A
	b.	What is the estimated auxiliary storage space used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
AT.2(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 3A CSCI LEVEL

AT.2(2)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.2(3)	a.	What is the total processing time allocated?	N/A
	b.	What is the estimated processing time used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
AT.3(1)	a.	What is the total I/O channel time allocated?	N/A
	b.	What is the estimated I/O channel time used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A

Note, for more than one I/O channel, list answers to a., b., and c. below and enter average of answers in boxes above.

I/O CHANNEL	a.	b.	c.

3A-6

METRIC WORKSHEET 3A CSCI LEVEL

AT.3(2) a. What is the total communication channel time allocated? [ ] N/A

b. What is the estimated communication channel time used (total less spare)? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

Note, for more than one communication channel, list answers to a., b., and c. below and enter average of answers in boxes above.

COMMUNICATION CHANNEL	a.	b.	c.
________	____	____	____
________	____	____	____
________	____	____	____
________	____	____	____

AU.1(2) a. How many estimated lines of source code, excluding comments? [ ] N/A

b. How many estimated lines of source code necessary to handle hardware and device interface protocol? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

AU.1(3) a. How many units? [ ] N/A

b. How many units perform processing of hardware and/or device interface protocol? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

AU.1(4) a. How much estimated processing time is typically spent executing the entire CSCI? [ ] N/A

	b.	How much estimated processing time is typically spent executing the hardware and device interface protocol units?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
CL.1(7)	a.	How many units receive inputs from other systems?	N/A
	b.	Calculate 1/a and enter score.	N/A
CL.1(8)	a.	How many units transmit outputs to other systems?	N/A
	b.	Calculate 1/a and enter score.	N/A
CL.2(1)		Do all data representations and translations comply with the established standard?	Y N N/A
CL.2(2)	a.	How many units perform data translations?	N/A
	b.	Calculate 1/a and enter score.	N/A
CP.1(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CP.1(2)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 3A | CSCI LEVEL

CP.1(3)	a.	How many data items are defined (i.e., documented with regard to source, meaning, and format)?	N/A
	b.	How many defined data items are referenced?	N/A
	c.	Calculate b/a and enter score.	N/A
CP.1(4)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CP.1(9)	a.	How many applicable units (answer of Y or N on 3B).	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CP.1(10)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CP.1(11)	a.	How many software problem reports have been recorded, to date?	N/A
	b.	How many recorded software problem reports have been closed (resolved), to date?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 3A CSCI LEVEL

CS.1(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.1(2)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.1(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.1(4)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.1(5)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.2(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 3A CSCI LEVEL

CS.2(2)	a.	How many applicable units (answer of Y or N on 3B)?	___ N/A
	b.	How many units with answer of Y (see 3B)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
CS.2(3)	a.	How many applicable units (answer of Y or N on 3B)?	___ N/A
	b.	How many units with answer of Y (see 3B)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
CS.2(6)	a.	How many applicable units (answer of Y or N on 3B)?	___ N/A
	b.	How many units with answer of Y (see 3B)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
DI.1(1)		Are graphic portrayals (e.g., figures, diagrams, tables) provided which identify the decomposition of all top-level CSC's to all lower-level CSC's and/or units?	Y N N/A
DO.1(1)		Are current versions of all software documentation related to the project free from access control (i.e., any member of the current project or other projects may access a copy of any document)?	Y N N/A
DO.2(1)		Is all the documentation structured and written clearly and simply such that procedures, functions, algorithms, etc. can be easily understood?	Y N N/A
DO.2(2)		Does the requirements/design documentation clearly depict control and data flow (e.g., graphic portrayal with accompanying explanations or PDL)?	Y N N/A
DO.2(3)		Does all documentation contain an indexing scheme which facilitates quickly locating and accessing various information in the document (e.g., hierarchical structured table of contents, inserted tabs)?	Y N N/A

DO.2(4) Do the software specifications and design and test documentation have separate volumes or separations within a single volume based on system functions, software functions, or software elements? Y N N/A

DO.2(7) Does the documentation contain comprehensive descriptions of all algorithms used and their limitations, including inputs, outputs, and required precision? Y N N/A

EP.1(2) a. How many applicable units (score entered on 3B)? ___ N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? ___ N/A

c. Calculate b/a and enter score. ___ N/A

EP.1(3) Have software units which are required to be optimized for processing efficiency been identified? Y N N/A

EP.1(4) a. How many applicable units (score entered on 3B)? ___ N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? ___ N/A

c. Calculate b/a and enter score. ___ N/A

EP.1(5) a. How many different overlays are used in this CSCI? ___ N/A

b. Calculate 1/a and enter score. ___ N/A

EP.1(6) a. How many applicable units (score entered on 3B)? ___ N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? ___ N/A

c. Calculate b/a and enter score. [ ] N/A

EP.2(2) Has the storage of all information (e.g., files, code, arrays, buffers) been organized for efficient processing (e.g., minimum search time)? Y N N/A

EP.2(4) a. How many applicable units (score entered on 3B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

EP.2(5) a. How many applicable units (score entered on 3B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

EP.2(6) Does the method(s) for relating similar data items facilitate efficient processing (e.g., arrays, doubly linked lists, directories)? Y N N/A

EP.2(7) a. How many applicable units (score entered on 3B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

ES.1(3) a. How many global variables? [ ] N/A

b. How many global variables are referenced by more than one name? [ ] N/A

/d2

c. Calculate 1-(b/a) and enter score. [ ] N/A

ES.1(4) Does the separation of the CSCI into segments (i.e., load modules) efficiently utilize the segmented memory space available (e.g., minimizing the largest CSCI segment length to minimize the memory segment size required for module execution)? Y N N/A

ES.1(6) a. How many applicable units (answer Y or N on 3B)? [ ] N/A

b. How many units with answer of Y (see 3B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

ES.1(8) Is the CSCI free from redundant storage of files and libraries (e.g., duplicate copies are not stored at different nodes in a network; multiple versions of the same file are not part of the working CSCI)? Y N N/A

FS.1(1) a. How many applicable units (score entered on 3B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

GE.1(1) a. How many units in total? [ ] N/A

b. How many units are called by more than one other unit? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

GE.2(1) a. How many units in this CSCI? [ ] N/A

	b.	In how many units are the following processing categories mixed: external input, external output, algorithmic processing?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
GE.2(2)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
GE.2(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
GE.2(4)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ID.1(1)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ID.1(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A

	c.	Calculate b/a and enter score.	[ ] N/A
ID.2(2)	a.	How many units in the CSCI?	[ ] N/A
	b.	How many units in the CSCI perform external input/output?	[ ] N/A
	c.	Calculate 1-(b/a) and enter score.	[ ] N/A
ID.2(3)	a.	How many units in the CSCI?	[ ] N/A
	b.	How many units in the CSCI contain operations dependent on word or character size?	[ ] N/A
	c.	Calculate 1-(b/a) and enter score.	[ ] N/A
ID.2(4)	a.	How many units in the CSCI?	[ ] N/A
	b.	How many units in the CSCI contain data element representations that are machine dependent?	[ ] N/A
	c.	Calculate 1-(b/a) and enter score.	[ ] N/A
MO.1(2)		Are all CSC's developed according to structured design techniques?	Y N N/A
MO.1(3)	a.	How many applicable units (answer of Y or N on 3B)?	[ ] N/A
	b.	How many units with answer of Y (see 3B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
MO.1(4)	a.	How many applicable units (score entered on 3B)?	[ ] N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	[ ] N/A

METRIC WORKSHEET 3A

CSCI LEVEL

	c.	Calculate b/a and enter score.	N/A
MO.1(5)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(6)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(7)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(8)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(9)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.2(2)	a.	How many interfaces among software units?	N/A

b. How many unit interfaces include:

b1. Content coupling [ ] N/A

b2. Common coupling [ ] N/A

b3. External coupling [ ] N/A

c. Calculate 1-((b1+b2+b3)/(3xa)) and enter score. [ ] N/A

MO.2(3) a. How many interfaces among software units? [ ] N/A

b. How many unit interfaces include:

b1. Control coupling [ ] N/A

b2. Stamp coupling [ ] N/A

b3. Data coupling [ ] N/A

c. Calculate ((b1+b2)/(2xa)+b3/a) and enter score. [ ] N/A

MO.2(5) a. How many applicable units (score entered on 3B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 3B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

OP.1(2) Are all specified error conditions reported to the operator/user such that the nature of the error and any response required by the operator/user are clearly identified and described in the error message? Y N N/A

3A-18

METRIC WORKSHEET 3A CSCI LEVEL

OP.1(3)		Is the capability provided for operator/user response to all reported errors as specified in the requirements?	Y N N/A
OP.1(10)	a.	How many total operator messages and responses are provided?	____ N/A
	b.	How many different format types are used for operator messages and responses?	____ N/A
	c.	Calculate 1-(b/a) and enter score.	____ N/A
SD.3(5)	a.	How many applicable units (answer of Y or N on 3B)?	____ N/A
	b.	How many units with answer of Y (see 3B)?	____ N/A
	c.	Calculate b/a and enter score.	____ N/A
SI.1(1)		Does the design of the CSCI reflect a structured design approach (e.g., top-down design)?	Y N N/A
SI.1(2)	a.	How many applicable units (answer of Y or N on 3B)?	____ N/A
	b.	How many units with answer of Y (see 3B)?	____ N/A
	c.	Calculate b/a and enter score.	____ N/A
SI.1(3)	a.	How many applicable units (answer of Y or N on 3B)?	____ N/A
	b.	How many units with answer of Y (see 3B)?	____ N/A
	c.	Calculate b/a and enter score.	____ N/A
SI.1(4)	a.	How many applicable units (answer of Y or N on 3B)?	____ N/A
	b.	How many units with answer of Y (see 3B)?	____ N/A

	c.	Calculate b/a and enter score.	N/A
SI.1(5)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(6)	a.	How many unique data items are in common blocks?	N/A
	b.	Calculate 1/a and enter score.	N/A
SI.1(7)	a.	How many unique data items are in common blocks?	N/A
	b.	How many unique common blocks?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(10)		Do the descriptions of all units identify all interfacing units and all interfacing hardware?	Y N N/A
SI.3(1)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 3A

CSCI LEVEL

SI.4(2)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(3)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(4)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(5)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(14)		Is repeated and redundant code avoided (e.g., through utilizing macros, procedures, and functions)?	Y N N/A
SI.5(1)	a.	How many applicable units (score entered on 3B)?	N/A

	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.5(2)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.5(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.6(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(1)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(2)	a.	How many applicable units (score entered on 3B)?	N/A

	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score	N/A
ST.1(3)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(4)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(5)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(6)	a.	How many units in this CSCI?	N/A
	b.	How many units modify the internal code or data of other units?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
ST.2(1)	a.	How many applicable units (score entered on 3B)?	N/A

	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.2(2)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.2(3)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.2(4)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.3(2)		Is I/O isolated from computation in the design?	Y N N/A
ST.3(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A

ST.3(4)	a.	How many units mix the management of primary and secondary storage resources with the management of data within the storage areas (e.g., executive unit that allocates storage for a process and controls what data can be accessed during process execution)?	[ ] N/A
	b.	Calculate 1/(1+a) and enter score.	[ ] N/A
ST.4(1)	a.	How many applicable units (score entered on 3B)?	[ ] N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
ST.4(2)	a.	How many applicable units (score entered on 3B)?	[ ] N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
ST.4(3)	a.	How many global data items are used in the CSCI?	[ ] N/A
	b.	How many global data items are modified by one unit and referenced by other units?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
ST.4(4)	a.	How many applicable units (answer of Y or N on 3B)?	[ ] N/A
	b.	How many units with answer of Y (see 3B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A

ST.4(5)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.5(1)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.5(2)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.5(3)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.5(4)	a.	How many applicable units (answer of Y or N on 3B)?	N/A
	b.	How many units with answer of Y (see 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
TC.1(1)		Does the description of each software unit identify all the specified requirements (at the top-level CSC or CSCI level) that the unit helps satisfy?	Y N N/A

TC.1(2)		Is the decomposition of all top-level CSC's into lower-level CSC's and software units graphically depicted?	Y N N/A
TN.1(1)		Are there lesson plans and training materials for operators, and users, and maintainers of the system (CSCI)?	Y N N/A
TN.1(2)		Are realistic simulation exercises provided for the system (CSCI)?	Y N N/A
TN.1(3)		Are "help" information and diagnostic information provided for the operator, end user, and maintainer of the system (CSCI) (e.g., an on-line list of legal commands or a list of the sequential steps in a process are provided)?	Y N N/A
TN.1(4)		Can system (CSCI) users select a level of aid and guidance according to their degree of expertise?	Y N N/A
VS.1(1)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
VS.1(2)	a.	How many applicable units (score entered on 3B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 3B)?	N/A
	c.	Calculate b/a and enter score.	N/A
VS.2(1)	a.	How many total interfaces are there between units in the CSCI?	N/A
	b.	How many unit interfaces are (to be) tested?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 3A CSCI LEVEL

VS.3(1)	Are all specified performance requirements (to be) tested?	Y N N/A
VS.3(2)	Are all units of this CSCI (to be) exercised during CSCI testing?	Y N N/A
VS.3(3)	Is there (to be) a summary table listing all test inputs and test outputs for the CSCI?	Y N N/A

GLOSSARY

Argument List: A list of data elements that specify the input and output parameters used during execution of a software unit.

Cohesion Value: The type of relationship that exists among the elements of each software entity (Function, CSCI, Unit). The following are relative values for seven types of cohesion.

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohension.

1) Coincidental
   - No meaningful relationships among the elements of an entity.
   - Difficult to describe the module's function(s).
2) Logical
   - Entity performs (at each invocation) one of a class of related functions (e.g., "edit all data").
   - Entity performs more than one function.
3) Classical
   - Entity performs one of a class of functions that are related in time (Program procedure).
   - Entity performs more than one function.
4) Procedural
   - Entity performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).

GLOSSARY (continued)

5) Communicational
- Entity has procedural strength; in addition, all of the elements "communicate" with one another (e.g., reference same data or pass data among themselves).
- All functions use the same data.

6) Informational
- Entity performs multiple functions where the functions (entry points in the entity) deal with a single data structure.
- Physical packaging together of two or more entities having functional strength.
- All functions use the same data.

7) Functional
- All entity elements are related to the performance of a single function.

Communication Channel: The pathways along which data/messages are communicated to the various system components or nodes (i.e., other computer, data storage units, special processors, etc.).

Control Elements: Any data items that select an operating mode or submode in the software unit, direct the sequential flow, or otherwise directly influence the function of the unit.

Control Variables: Any data items that select an operating mode or submode in the software unit, direct the sequential flow, or otherwise directly influence the function of the unit.

Coupling: The type of relationship that exists between two software entities (Functions, CSCIs, Units). In achieving a highly modular design it is essential to minimize the relationships among software entities. The goal is to design software entities with low coupling. The scae of coupling from worst to best is: (1) Content Coupling, 2) Common Coupling, 3) External Coupling, 4) Control Coupling, 5) Stamp Coupling, and 6) Data Coupling.

GLOSSARY (continued)

1) Content Coupling - One software entity references the contents of another software entity.

2) Common Coupling - Software entities reference a shared global data structure.

3) External Coupling - Software entities reference the same externally declared symbol.

4) Control Coupling - One software entity passes control elements as arguments to another software entity.

5) Stamp Coupling - Two software entities reference the same data structure, which is not global.

6) Data Coupling - One software entity calls another and the software entities are not coupled as defined above (in 1 through 5).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Reference: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Database Management Scheme: The methods and commands used to access or operate the database management software system.

Design Representation: A formal statement of the details or organization of a design using one of a number of design representation methodologies, such as, Flow Charts, HIPO Charts, PDL, etc.

Halstead's Level of Difficulty: The metric is based on Halstead's concept of the level of difficulty. A program with a high value of difficulty is likey to be more difficult to

GLOSSARY (continued)

construct and this may lead to more errors. The level of difficulty is a measure of "error-proneness". Programming difficulty increases if additional operators are introduced and if an operand is used repetitively.

I/O Channel: The pathways along which data/messages are communicated to the various user-oriented peripherals in the system (CRT, Printer).

Lines of Code: The number of lines of source code, excluding comment lines and blank lines.

Microcode Instructions: A low-level computer instruction specifying a single machine operation.

Multiple Transfer Index Parameter: A value used to select a variation in the order of code execution (i.e., case statement, program switch, etc.).

Range-Test: A test performed to validate the object of interest over the complete spectrum of applicable values.

Subscript Value: A value used to reference an entity from a group of related objects (i.e., table index, array index, etc.).

WORKSHEET PREPARATION INSTRUCTIONS

SECTION A - GENERAL INFORMATION

1. Enter the name and contract number of the project.
2. Enter the date the worksheet is prepared (month, day, year).
3. Enter the name of the person responsible for completing the worksheet.
4. Enter the name of the entity to which the worksheet is applied.
5. Enter the number and name of all documentation which is used as source material for answering the metric questions in Section B.
6. Enter comments reflecting the inspector's observations on product quality and any additional information regarding specific metric questions. Attach additional sheets, as necessary.

SECTION B - METRIC QUESTIONS

Answer all applicable metric questions by circling the appropriate answer (Y = yes, N = no, N/A = not applicable) or by entering the appropriate value. A glossary of terms is provided at the end of this section.

ふ) レ

METRIC WORKSHEET 3B UNIT LEVEL

SECTION A - GENERAL INFORMATION

1. PROJECT ______________________________
2. DATE ______________________________
3. INSPECTOR ______________________________
4. PRODUCT ______________________________
5. SOURCE DOCUMENTATION:

____
____
____
____
____
____
____

6. INSPECTOR COMMENTS:

3B-2

SECTION B - METRIC QUESTIONS

AM.1(3)	d.	When an error condition is detected, is resolution of the error determined by the calling unit?	Y N N/A
AM.2(7)	d.	Is a check performed before processing begins to determine that all data is available?	Y N N/A
AP.1(1)	d.	Is this unit free from specific references to the data base management scheme (e.g., all data calls for data base information are processed through an executive)?	Y N N/A
AP.2(1)	d.	How many parameters in the argument list for the unit?	___ N/A
	e.	How many global variables are referenced by the unit?	___ N/A
	f.	Calculate d/(d+e) and enter score.	___ N/A
AP.2(2)	d.	Do the comments for global data explain where the data is derived, the data's composition, and how the data is used?	Y N N/A
AP.2(3)	d.	Is the logical processing free from specific data storage and retrieval references (e.g., data symbolically defined and referenced)?	Y N N/A
AP.2(4)	d.	Do the comments for all parameter input and output and local variables explain the composition and use of each data item?	Y N N/A
AP.3(1)	d.	Is the unit free from specific references to the computer architecture?	Y N N/A
AP.4(1)	d.	Is the unit free from microcode instruction statements?	Y N N/A
AT.1(1)	d.	Are all variable dimensions and sizes of dynamic arrays defined parametrically? (Note that this question must be answered N rather than N/A where the language does not provide for parametric definition.)	Y N N/A

AT.2(1)	d.	Are all accuracy, convergence, timing attributes, and timing limitations defined parametrically?	Y N N/A
AT.2(2)	d.	Are tables used in a manner which would ease the task of changing or expanding capability?	Y N N/A
CP.1(1)	d.	Are all inputs, processing, and outputs clearly and precisely defined?	Y N N/A
CP.1(2)	d.	How many data references are identified?	___ N/A
	e.	How many identified data references are documented with regard to source, meaning, and format?	___ N/A
	f.	Calculate e/d and enter score.	___ N/A
CP.1(4)	d.	How many data references are identified?	___ N/A
	e.	How many identified data references are computed or obtained from an external source (e.g., referencing global data with preassigned values, input parameters with preassigned values)?	___ N/A
	f.	Calculate e/d and enter score.	___ N/A
CP.1(9)	d.	Are all conditions and alternative processing options defined for each decision point?	Y N N/A
CP.1(10)	d.	Are all parameters in the argument list used?	Y N N/A
CS.1(1)	d.	Are all design representations in the formats of the established standard?	Y N N/A
CS.1(2)	d.	Does the calling sequence protocol (between units) comply with established standard?	Y N N/A
CS.1(3)	d.	Does the I/O protocol and format comply with the established standard?	Y N N/A

CS.1(4)	d.	Does the handling of errors comply with the established standard?	Y N N/A
CS.1(5)	d.	Do all references to this unit use the same, unique name?	Y N N/A
CS.2(1)	d.	Does all data representation comply with the established standard?	Y N N/A
CS.2(2)	d.	Does the naming of all data comply with the established standard?	Y N N/A
CS.2(3)	d.	Is the definition and use of all global variables in accordance with the established standard?	Y N N/A
CS.2(6)	d.	Do all references to the same data use a single, unique name?	Y N N/A
EP.1(2)	d.	How many loops in this unit (while, repeat until, and iteration loops)?	N/A
	e.	How many loops contain non-loop dependent statements (e.g., initializing a non-loop dependent variable)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.1(4)	d.	How many instances of two or more operations in an expression (i.e., compound expression)?	N/A
	e.	How many compound expressions are recalculated needlessly (all variables in the expression have not been reassigned values)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.1(6)	d.	How many instances of bit/byte packing/unpacking are performed?	N/A
	e.	How many instances of bit/byte packing/unpacking are performed needlessly within a loop (could be performed outside the loop)?	N/A

	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
EP.2(4)	d.	How many arithmetic expressions?	[ ] N/A
	e.	How many arithmetic expressions with different sized components in the same expression (e.g., byte/word/double word)?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
EP.2(5)	d.	How many arithmetic expressions?	[ ] N/A
	e.	How many arithmetic expressions with mixed data types in the same expression (e.g., integer/real/boolean/literal)?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
EP.2(7)	d.	How many data items (e.g., arrays, constants, variables)?	[ ] N/A
	e.	How many data items are modified?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
ES.1(6)	d.	Are there any data packing operations?	Y N N/A
FS.1(1)	d.	Does this unit perform a single function?	Y N N/A
GE.2(2)	d.	Is this unit free from machine-dependent operations?	Y N N/A
GE.2(3)	d.	Is this unit free from strict limitations on the volume of data items it processes (e.g., data volume limits are parameterized)?	Y N N/A
GE.2(4)	d.	Is this unit free from strict limitations on the values of input data (e.g., no error tolerances are specified; no range tests or reasonableness checks are performed)?	Y N N/A

ID.1(1) d. How many estimated lines of code, excluding comments? [ ] N/A

e. How many references to system library routines, utilities, or other system-provided facilities? [ ] N/A

f. Calculate 1-(e/d) and enter score. [ ] N/A

ID.1(3) d. Is the software free from using any non-standard constructs of the implementation language(s)? Y N N/A

MO.1(3) d. Are the estimated lines of source code for this unit 100 lines or less, excluding comments? Y N N/A

MO.1(4) d. How many parameters are there in the calling sequence? [ ] N/A

e. How many calling sequence parameters are control variables (e.g., select an operating mode or submode, direct the sequential flow, directly influence the function of the software)? [ ] N/A

f. Calculate e/d and enter score. [ ] N/A

MO.1(5) d. Is all input data passed into the unit through calling sequence parameters (i.e., no data is input through global areas or input statements)? Y N N/A

MO.1(6) d. Is output data passed back to the calling unit (e.g., through calling sequence parameters or global areas)? Y N N/A

MO.1(7) d. Is control always returned to the calling unit when execution is completed? Y N N/A

MO.1(8) d. Is temporary storage (i.e., workspace reserved for intermediate or partial results used only by this unit during execution (i.e., is not shared with other units)? Y N N/A

MO.1(9)	d.	Does this unit have a single processing objective (i.e., all processing within this unit is related to the same objective)?	Y N N/A
MO.2(5)	d.	What is the cohesion value of this unit?	___ N/A
SD.3(5)	d.	Is the unit structured in the standard established format?	Y N N/A
SI.1(2)	d.	Is the unit independent of the source of the input and the destination of the output?	Y N N/A
SI.1(3)	d.	Is the unit independent of the knowledge of prior processing?	Y N N/A
SI.1(4)	d.	Does the unit description/prologue include input, output, processing, and limitations?	Y N N/A
SI.1(5)	d.	How many entrances into the unit?	___ N/A
	e.	How many exits from the unit?	___ N/A
	f.	Calculate (1/d+1/e)(x½) and enter score.	___ N/A
SI.3(1)	d.	How many conditional branch statements are there (e.g., IF, WHILE, REPEAT, DO/FOR LOOP, CASE)?	___ N/A
	e.	How many unconditional branch statements are there (e.g., GO TO, CALL, RETURN)?	___ N/A
	f.	Calculate 1/(1+d+e) and enter score.	___ N/A
SI.4(1)	d.	Is the flow of control from top to bottom (i.e., flow of control does not jump erratically)?	Y N N/A
SI.4(2)	d.	How many estimated lines of source code, excluding comments?	___ N/A

3B-8

	e.	How many negative boolean and compound boolean expressions are used?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
SI.4(3)	d.	How many loops (e.g., WHILE, DO/FOR, REPEAT)?	[ ] N/A
	e.	How many loops with unnatural exits (e.g., jumps out of loop, return statement)?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
SI.4(4)	d.	In many iteration loops (i.e., DO/FOR loops)?	[ ] N/A
	e.	In how many iteration loops are indices modified to alter the fundamental processing of the loop?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
SI.4(5)	d.	Is the unit free from all self-modification of code (i.e., does not alter instructions, overlays of code, etc.)?	Y N N/A
SI.5(1)	d.	How many data items are used as input?	[ ] N/A
	e.	Calculate 1/(1+d) and enter score.	[ ] N/A
SI.5(2)	d.	How many data items are used for output?	[ ] N/A
	e.	How many parameters in the units calling sequence return output values?	[ ] N/A
	f.	Calculate e/f and enter score.	[ ] N/A
SI.5(3)	d.	Does the unit perform a single, nondivisible function?	Y N N/A

METRIC WORKSHEET 3B | UNIT LEVEL

SI.6(1)	d.	How many unique operations?	[ ] N/A
	e.	How many unique operands?	[ ] N/A
	f.	How many total operands?	[ ] N/A
	g.	Calculate 1-(2xe)/(dxf) and enter score.	[ ] N/A
ST.1(1)	d.	How many data items are in this unit's interface (i.e., data items used to input or output data)?	[ ] N/A
	e.	Calculate 1/(1+d) and enter score.	[ ] N/A
ST.1(2)	d.	How many global data items are in this unit's interface are not adequately commented (i.e., lack comments regarding the purpose, type, or limitations)?	[ ] N/A
	e.	Calculate 1/d and enter score.	[ ] N/A
ST.1(3)	d.	How many data items are in the unit's interface?	[ ] N/A
	e.	How many interface data items are in the unit with negative qualification logic (e.g., boolean values that return "TRUE" upon failure rather than success)?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
ST.1(4)	d.	How many data items are in this unit's interface?	[ ] N/A
	e.	Calculate 1/(1+d) and enter score.	[ ] N/A
ST.1(5)	d.	Is the unit interface established solely by arguments in the calling sequence parameter list?	Y N N/A

ST.2(1)	d.	How many unique execution paths are in the unit?	__ N/A
	e.	Calculate 1/d and enter score.	__ N/A
ST.2(2)	d.	How many conditional branch statements are there (e.g., IF, WHILE, REPEAT, CASE)?	__ N/A
	e.	Calculate 1/(1+d) and enter score.	__ N/A
ST.2(3)	d.	How many other units are called by this unit (e.g., calls to other functions, subroutines, and procedures)?	__ N/A
	e.	Calculate 1/(1+d) and enter score.	__ N/A
ST.2(4)	d.	How many iteration loops are there in the unit (e.g., DO/FOR loops)?	__ N/A
	e.	Calculate 1/(1+d) and enter score.	__ N/A
ST.3(3)	d.	Is temporary storage (i.e., workspace reserved intermediate or partial results) used only by this unit during execution (i.e., is not shared with other units)?	Y N N/A
ST.4(1)	d.	How many global data items are used in the unit?	__ N/A
	e.	How many parameters are in this unit's calling sequence parameter list?	__ N/A
	f.	Calculate e/d and enter score.	__ N/A
ST.4(2)	d.	How many global data items are used in this unit?	__ N/A
	e.	Calculate 1/(1+d) and enter score.	__ N/A

ST.4(4)	d.	Does this unit have a single entrance (i.e., all units calling this unit must enter at the same location)?	Y N N/A
ST.4(5)	d.	Does this unit's communication with all interfacing units pass only data parameters (i.e., does not pass any control elements)?	Y N N/A
ST.5(1)	d.	Is the unit free from unnecessarily recomputing the same value?	Y N N/A
ST.5(2)	d.	Is the unit free from statements which are never executed?	Y N N/A
ST.5(3)	d.	Is the meaning of each data item consistent throughout the unit (i.e., the use associated with each data item does not change)?	Y N N/A
ST.5(4)	d.	Is the unit free from unnecessary intermediate data items?	Y N N/A
VS.1(1)	d.	How many execution paths are there?	N/A
	e.	How many execution paths are (to be) tested?	N/A
	f.	Calculate e/d and enter score.	N/A
VS.1(2)	d.	How many total input parameters are there?	N/A
	e.	How many input parameters are to be tested?	N/A
	f.	Calculate e/d and enter score.	N/A

GLOSSARY

Argument List: A list of data elements that specify the input and output parameters used during execution of a software unit.

Cohesion Value: The type of relationship that exists among the elements of each software entity (Function, CSCI, Unit).

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohension.

1) Coincidental
   - No meaningful relationships among the elements of an entity.
   - Difficult to describe the module's function(s).
2) Logical
   - Entity performs (at each invocation) one of a class of related functions (e.g., "edit all data").
   - Entity performs more than one function.
3) Classical
   - Entity performs one of a class of functions that are related in time (Program procedure).
   - Entity performs more than one function.
4) Procedural
   - Entity performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).

GLOSSARY (continued)

5) Communicational

- Entity has procedural strength; in addition, all of the elements "communicate" with one another (e.g., reference same data or pass data among themselves).
- All functions use the same data.

6) Informational

- Entity performs multiple functions where the functions (entry points in the module) deal with a single data structure.
- Physical packaging together of two or more entities having functional strength.
- All functions use the same data.

7) Functional

- All entity elements are related to the performance of a single function.

Communication Channel: The pathways along which data/messages are communicated to the various system components or nodes (i.e., other computer, data storage units, special processors, etc.).

Control Element: Any data items that select an operating mode or submode in the software unit, direct the sequential flow, or otherwise directly influence the function of the unit.

Control Variables: Any data items that select an operating mode or submode in the software unit, direct the sequential flow, or otherwise directly influence the function of the unit.

Coupling: The type of relationship that exists between two software entities (Functions, CSCIs, Units). In achieving a highly modular design it is essential to minimize the relationships among software entities. The goal is to design software entities with low coupling. The scae of coupling from worst to best is: (1) Content Coupling, 2) Common Coupling, 3) External Coupling, 4) Control Coupling, 5) Stamp Coupling, and 6) Data Coupling.

GLOSSARY (continued)

1) Content Coupling - One software entity references the contents of another software entity.

2) Common Coupling - Software entities reference a shared global data structure.

3) External Coupling - Software entities reference the same externally declared symbol.

4) Control Coupling - One software entity passes control elements as arguments to another software entity.

5) Stamp Coupling - Two software entities reference the same data structure, which is not global.

6) Data Coupling - One software entity calls another and the software entities are not coupled as defined above (in 1 through 5).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Reference: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Database Management Scheme: The methods and commands used to access or operate the database management software system.

Design Representation: A formal statement of the details or organization of a design using one of a number of design representation methodologies, such as, Flow Charts, HIPO Charts, PDL, etc.

Halstead's Level of Difficulty: The metric is based on Halstead's concept of the level of difficulty. A program with a high value of difficulty is likey to be more difficult to construct and this may lead to more errors. The level of difficulty is a measure of

GLOSSARY (continued)

"error-proneness". Programming difficulty increases if additional operators are introduced and if an operand is used repetitively.

I/O Channel: The pathways along which data/messages are communicated to the various user-oriented peripherals in the system (CRT, Printer).

Lines of Code: The number of lines of source code, excluding comment lines and blank lines.

Microcode Instructions: A low-level computer instruction specifying a single machine operation.

Multiple Transfer Index Parameter: A value used to select a variation in the order of code execution (i.e., case statement, program switch, etc.).

Range-Test: A test performed to validate the object of interest over the complete spectrum of applicable values.

Subscript Value: A value used to reference an entity from a group of related objects (i.e., table index, array index, etc.).

WORKSHEET PREPARATION INSTRUCTIONS

SECTION A - GENERAL INFORMATION

1. Enter the name and contract number of the project.
2. Enter the date the worksheet is prepared (month, day, year).
3. Enter the name of the person responsible for completing the worksheet.
4. Enter the name of the entity to which the worksheet is applied.
5. Enter the number and name of all documentation which is used as source material for answering the metric questions in Section B.
6. Enter comments reflecting the inspector's observations on product quality and any additional information regarding specific metric questions. Attach additional sheets, as necessary.

SECTION B - METRIC QUESTIONS

Answer all applicable metric questions by circling the appropriate answer (Y = yes, N = no, N/A = not applicable) or by entering the appropriate value. A glossary of terms is provided at the end of this section.

METRIC WORKSHEET 4A CSCI LEVEL

SECTION A - GENERAL INFORMATION

1. PROJECT ____________________
2. DATE ____________________
3. INSPECTOR ____________________
4. PRODUCT ____________________
5. SOURCE DOCUMENTATION:

____

____

____

____

____

____

____

6. INSPECTOR COMMENTS:

4A-2

SECTION B - METRIC QUESTIONS

AC.1(8) Do the outputs associated with applicable functions (e.g., mission critical functions) provide enough precision to support accuracy objectives? Y N N/A

AM.1(3) a. How many applicable units (answer of Y or N on 4B)? ____ N/A

b. How many units with answer of Y (see 4B)? ____ N/A

c. Calculate b/a and enter score. ____ N/A

AM.2(2) Are values of all applicable external inputs with range specifications checked with respect to specified range prior to use? Y N N/A

AM.2(3) Are all applicable external inputs checked with respect to specified conflicting requests prior to use? Y N N/A

AM.2(4) Are all applicable external inputs checked with respect to specified illegal combinations prior to use? Y N N/A

AM.2(5) Are all applicable external inputs checked for reasonableness before processing begins? Y N N/A

AM.2(6) Are all detected errors, with respect to applicable external inputs, reported before processing begins? Y N N/A

AM.2(7) a. How many applicable units (answers of Y or N on 4B) ____ N/A

b. How many units with answer of Y (see 4B) ____ N/A

c. Calculate b/a and enter score. ____ N/A

AM.3(2) Are all critical loop and multiple transfer index parameters (e.g., supporting a mission critical function) checked for out-of-range values before use? Y N N/A

AM.3(3)		Are all critical subscripts (e.g., supporting a mission critical function) checked for out-of-range values before use?	Y N N/A
AM.3(4)		Are all critical output data (e.g., supporting a mission critical function) checked for reasonable values prior to final outputting?	Y N N/A
AP.1(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.2(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of (see 4B)?	N/A

	c.	Calculate b/a and enter score.	N/A
AP.3(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.3(2)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AP.4(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.1(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.1(2)	a.	What is the total memory space allocated?	N/A
	b.	What is the actual memory space used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
AT.1(3)	a.	What is the total auxiliary storage space allocated?	N/A

	b.	What is the actual auxiliary storage space used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
AT.2(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.2(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
AT.2(3)	a.	What is the total processing time allocated?	N/A
	b.	What is the actual processing time used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
AT.3(1)	a.	What is the total I/O channel time allocated?	N/A
	b.	What is the actual channel time used (total less spare)?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A

Note, for more than one I/O channel, list answers to a., b., and c. below and enter average of answers in boxes above.

I/O CHANNEL	a.	b.	c.

METRIC WORKSHEET 4A CSCI LEVEL

AT.3(2) a. What is the total communication channel time allocated? [ ] N/A

b. What is the actual communication channel time used (total less spare)? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

Note, for more than one communication channel, list answers to a., b., and c. below and enter average of answers in boxes above.

COMMUNICATION CHANNEL	a.	b.	c.

AU.1(2) a. How many lines of source code, excluding comments? [ ] N/A

b. How many lines of source code necessary to handle hardware and device interface protocol? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

AU.1(3) a. How many units? [ ] N/A

b. How many units perform processing of hardware and/or device interface protocol? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

AU.1(4) a. How much processing time is typically spent executing the entire CSCI? [ ] N/A

b. How much processing time is typically spent executing the hardware and device interface protocol units? [ ] N/A

c. Calculate 1-(b/a) and enter score. [ ] N/A

CL.1(7) a. How many units receive inputs from other systems? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

CL.1(8) a. How many units transmit outputs to other systems? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

CL.2(1) Do all data representations and translations comply with the established standard? Y N N/A

CL.2(2) a. How many units perform data translations? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

CP.1(2) a. How many applicable units (score entered on 4B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 4B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

CP.1(3) a. How many data items are defined (i.e., documented with regard to source, meaning, and format)? [ ] N/A

b. How many defined data items are referenced? [ ] N/A

	c.	Calculate b/a and enter score.	[ ] N/A
CP.1(4)	a.	How many applicable units (score entered on 4B)?	[ ] N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
CP.1(9)	a.	How many applicable units (answer of Y or N on 4B)?	[ ] N/A
	b.	How many units with answer of Y (see 4B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
CP.1(10)	a.	How many applicable units (answer of Y or N on 4B)?	[ ] N/A
	b.	How many units with answer of Y (see 4B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
CP.1(11)	a.	How many software problem reports have been recorded, to date?	[ ] N/A
	b.	How many recorded software problem reports have been closed (resolved), to date?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A

METRIC WORKSHEET 4A | CSCI LEVEL

CS.1(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.1(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.1(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.1(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.2(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
CS.2(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A

4A-10

METRIC WORKSHEET 4A

CSCI LEVEL

CS.2(3)	a.	How many applicable units (answer of Y or N on 4B)?	___ N/A
	b.	How many units with answer of Y (see 4B)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
CS.2(6)	a.	How many applicable units (answer of Y or N on 4B)?	___ N/A
	b.	How many units with answer of Y (see 4B)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
DO.1(1)		Are current versions of all software documentation related to the project free from access control (i.e., any member of the current project or other projects may access a copy of any document)?	Y N N/A
DO.2(1)		Is all the documentation structured and written clearly and simply such that procedures, functions, algorithms, etc. can be easily understood?	Y N N/A
DO.2(3)		Does each document contain an indexing scheme which facilitates quickly locating and accessing various information in the document (e.g., hierarchical structured table of contents, inserted tabs)?	Y N N/A
DO.2(4)		Do the software specifications and design and test documentation have separate volumes or separation within a single volume based on system functions, software functions, or software elements?	Y N N/A
DO.2(5)		Are all the software listings included in the software documentation?	Y N N/A
EP.1(2)	a.	How many applicable units (score entered on 3B)?	___ N/A
	b.	What is total score for all applicable units (add applicable	

unit scores from 3B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

EP.1(3) a. How many applicable units (score entered on 4B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 4B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

EP.1(4) a. How many applicable units (score entered on 4B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 4B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

EP.1(5) a. How many different overlays are used in this CSCI? [ ] N/A

b. Calculate 1/a and enter score. [ ] N/A

EP.1(6) a. How many applicable units (score entered on 4B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 4B)? [ ] N/A

c. Calculate b/a and enter score. [ ] N/A

EP.2(4) a. How many applicable units (score entered on 4B)? [ ] N/A

b. What is total score for all applicable units (add applicable unit scores from 4B)? [ ] N/A

	c.	Calculate b/a and enter score.	[ ] N/A
EP.2(5)	a.	How many applicable units (score entered on 4B)?	[ ] N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
EP.2(7)	a.	How many applicable units (score entered on 4B)?	[ ] N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
ES.1(3)	a.	How many global variables?	[ ] N/A
	b.	How many global variables are referenced by more than one name?	[ ] N/A
	c.	Calculate 1-(b/a) and enter score.	[ ] N/A
ES.1(4)		Does the separation of the CSCI into segments (i.e., load modules) efficiently utilize the segmented memory space available (e.g., minimizing the largest CSCI segment length to minimize the memory segment size required for module execution)?	Y N N/A
ES.1(6)	a.	How many applicable units (answer of Y or N on 4B)?	[ ] N/A
	b.	How many units with answer of Y (see 4B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A

METRIC WORKSHEET 4A — CSCI LEVEL

ES.1(7)	a.	How many total software units?	☐ N/A
	b.	How many software units are optimized for storage efficiency (i.e., compiled with a storage optimizing compiler or coded in assembly language)?	☐ N/A
	c.	Calculate 1-(b/a) and enter score.	☐ N/A
ES.1(8)		Is the CSCI free from redundant storage of files and libraries (e.g., duplicate copies at different nodes in a network; multiple versions of the same file are not part of the working CSCI)?	Y N N/A
FS.1(1)	a.	How many applicable units (answer of Y or N on 4B)?	☐ N/A
	b.	How many units with answer of Y (see 4B)?	☐ N/A
	c.	Calculate b/a and enter score.	Y N N/A
FS.1(2)	a.	How many applicable units (answer of Y or N on 4B)?	☐ N/A
	b.	How many units with answer of Y (see 4B)?	☐ N/A
	c.	Calculate b/a and enter score.	☐ N/A
GE.1(1)	a.	How many units in total?	☐ N/A
	b.	How many units are called by more than one other unit?	☐ N/A
	c.	Calculate b/a and enter score.	☐ N/A
GE.2(1)	a.	How many units in this CSCI?	☐ N/A
	b.	In how many units are the following processing categories mixed: external input, external output, algorithmic processing?	☐ N/A

4A-14

	c.	Calculate 1-(b/a) and enter score.	N/A
GE.2(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
GE.2(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
GE.2(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ID.1(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ID.1(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ID.2(2)	a.	How many units in the CSCI?	N/A

	b.	How many units in the CSCI perform external input/output?	___ N/A
	c.	Calculate 1-(b/a) and enter score.	___ N/A
ID.2(3)	a.	How many units in the CSCI?	___ N/A
	b.	How many units in the CSCI contain operations dependent on word or character size?	___ N/A
	c.	Calculate 1-(b/a) and enter score.	___ N/A
ID.2(4)	a.	How many units in the CSCI?	___ N/A
	b.	How many units in the CSCI contain data element representations that are machine dependent?	___ N/A
	c.	Calculate 1-(b/a) and enter score.	___ N/A
MO.1(2)		Are all units coded and tested according to structural techniques?	Y N N/A
MO.1(3)	a.	How many applicable units (answer of Y or N on 4B)?	___ N/A
	b.	How many units with answer of Y (see 4B)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
MO.1(4)	a.	How many applicable units (answer of Y or N on 4B)?	___ N/A
	b.	How many units with answer of Y (see 4B)?	___ N/A
	c.	Calculate b/a and enter score.	___ N/A
MO.1(5)	a.	How many applicable units (answer of Y or N on 4B)?	___ N/A

	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(6)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(7)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(8)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.1(9)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
MO.2(2)	a.	How many interfaces among software units?	N/A
	b.	How many unit interfaces include:	
		b1. content coupling	N/A
		b2. common coupling	N/A

		b3. external coupling	[ ] N/A
	c.	Calculate 1-((b1+b2+b3)/(3xa)) and enter score.	[ ] N/A
MO.2(3)	a.	How many interfaces among software units?	[ ] N/A
	b.	How many unit interfaces include:	
		b1. control coupling	[ ] N/A
		b2. stamp coupling	[ ] N/A
		b3. data coupling	[ ] N/A
	c.	Calculate ((b1+b2)/(2xa)+(b3/a)) and enter score.	[ ] N/A
MO.2(5)	a.	How many applicable units (score entered on 4B)?	[ ] N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	[ ] N/A
	c.	Calculate b/a and enter score.	[ ] N/A
OP.1(2)		Are all specified error conditions reported to the operator/user such that the nature of the error and any response required by the operator/user are clearly identified and described in the error message?	Y N N/A
OP.1(3)		Is the capability provided for operator/user response to all reported errors as specified in the requirements?	Y N N/A
OP.1(10)	a.	How many total operator messages and responses are provided?	[ ] N/A
	b.	How many different format types are used for operator messages and responses?	[ ] N/A

	c.	Calculate 1-(b/a) and enter score.	N/A
SD.1(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A

	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(6)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(7)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.2(8)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.3(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.3(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.3(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A

	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.3(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.3(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SD.3(6)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A

	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(6)	a.	How many unique data items are in common blocks?	N/A
	b.	Calculate 1/a and enter score.	N/A
SI.1(7)	a.	How many unique data items are in common blocks?	N/A
	b.	How many unique common blocks?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.1(10)		Do the descriptions of all units identify all interfacing units and all interfacing hardware?	Y N N/A
SI.2(1)	a.	How many units in total?	N/A
	b.	How many units are implemented in a structural language or using a preprocessor?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.3(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A

	c.	Calculate b/a and enter score.	N/A
SI.4(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A

SI.4(6)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(7)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(8)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(9)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(10)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A

SI.4(11)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(12)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(13)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.4(14)		Is repeated and redundant code avoided (e.g., through utilizing macros, procedures, and functions)?	Y N N/A
SI.5(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.5(2)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A

METRIC WORKSHEET 4A　　　　CSCI LEVEL

SI.5(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
SI.6(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(2)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(3)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(4)	a.	How many applicable units (score entered on 4B)?	N/A

	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.1(6)	a.	How many units in this CSCI?	N/A
	b.	How many units modify the internal code or data of other units?	N/A
	c.	Calculate 1-(b/a) and enter score.	N/A
ST.2(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.2(2)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.2(3)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A

	c.	Calculate b/a and enter score.	N/A
ST.2(4)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.2(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.3(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.3(4)	a.	How many units mix the management of primary and secondary storage resources with the management of data within the storage areas (e.g., executive unit that allocates storage for a process and controls what data can be accessed during process execution)?	N/A
	b.	Calculate 1/(1+a) and enter score.	N/A
ST.4(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A

ST.4(2)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.4(3)	a.	How many global data items are used in the CSCI?	N/A
	b.	How many global data items are modified by one unit and referenced by other units?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.4(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.4(5)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.5(1)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.5(2)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A

	c.	Calculate b/a and enter score.	N/A
ST.5(3)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
ST.5(4)	a.	How many applicable units (answer of Y or N on 4B)?	N/A
	b.	How many units with answer of Y (see 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
VS.1(1)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A
VS.1(2)	a.	How many applicable units (score entered on 4B)?	N/A
	b.	What is total score for all applicable units (add applicable unit scores from 4B)?	N/A
	c.	Calculate b/a and enter score.	N/A

GLOSSARY

Argument List: A list of data elements that specify the input and output parameters used during execution of a software unit.

Cohesion Value. The type of relationship that exists among the elements of each software entity (Function, CSCI, Unit). The following are relative values for seven types ofcohesion.

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohesion.

1) Coincidental
   - o No meaningful relationships among the elements of an entity
   - o Difficult to describe the module's function(s).

2) Logical
   - o Entity (at each invocation) one of a class of related functions (e.g., "edit all data").
   - o Entity performs more than one function.

3) Classical
   - o Entity performs one of a class of functions that are related in time (Program procedure).
   - o Entity performs more than one function.

4) Procedural
   - o Entity performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).

GLOSSARY (Continued)

5) Communicational
   - o Entity has procedural strength; in addition, all of the elements "communicate" with one other (e.g., reference same data or pass data among themselves).
   - o All functions use the same data.

6) Informational
   - o Entity performs multiple functions where the functions (entry points in the unit) deal with a single data structure.
   - o Physical packaging together of two or more entities having functional strength.
   - o All functions use the same data.

7) Functional
   - o All entity elements are related to the performance of a single function.

Control Elements: Any data items that select an operating mode or submode in the Software unit, direct the sequential flow, or otherwise directly influence the function of the unit.

Control Variables: Any data items that select an operating mode or submode in the software unit, direct the sequential flow, or otherwise directly influence the function of the unit.

Coupling: The type of relationship that exists between two software entities (Functions, CSCIs, Units). In achieving a highly modular design it is essential to minimize the relationships among software entities. The goal is to design software entities with low coupling. The scale of coupling from worst to best is: 1) Content Coupling, 2) Common Coupling, 3) External Coupling, 4) Control Coupling, 5) Stamp Coupling, and 6) Data Coupling.

1) Content Coupling - One software entity references the contents of another software entity.

GLOSSARY (Continued)

2) Common Coupling - Software entities reference a shared global data structure.

3) External Coupling - Software entities reference the same externally declared symbol.

4) Control Coupling - One software entity passes control elements as arguments to another software entity.

5) Stamp Coupling - Two software entities reference the same data structure, which is not global.

6) Data Coupling - One software entity calls another and the software entities are not coupled as defined above (in 1 through 5).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Reference: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Design Representation: A formal statement of the details or organization of a design using one of a number of design representation methodologies, such as, Flow Charts, HIPO Charts, PDL, etc.

Halstead's Level of Difficulty: The metric is based on Halstead's concept of the level of difficulty. A program with a high value of difficulty is likely to be more difficult to construct and this may lead to more errors. The level of difficulty is a measure of "error-proneness". Programming difficulty increases if additional operators are introduced and if an operand is used repetitively.

Lines of Code: The number of lines of source code, excluding comment lines and blank lines.

GLOSSARY (Continued)

Microcode Instructions: A low-level computer instruction specifying a single machine operation.

Multiple Transfer Index Parameter: A value used to select a variation in the order of code execution (i.e., case statement, program switch, etc.).

Range-Test: A test performed to validate the object of interest over the complete spectrum of applicable values.

Subscript Value: A value used to reference an entity from a group of related objects (i.e., table index, array index, etc.).

4A-34

WORKSHEET PREPARATION INSTRUCTIONS

SECTION A - GENERAL INFORMATION

1. Enter the name and contract number of the project.
2. Enter the date the worksheet is prepared (month, day, year).
3. Enter the name of the person responsible for completing the worksheet.
4. Enter the name of the entity to which the worksheet is applied.
5. Enter the number and name of all documentation which is used as source material for answering the metric questions in Section B.
6. Enter comments reflecting the inspector's observations on product quality and any additional information regarding specific metric questions. Attach additional sheets, as necessary.

SECTION B - METRIC QUESTIONS

Answer all applicable metric questions by circling the appropriate answer (Y = yes, N = no, N/A = not applicable) or by entering the appropriate value. A glossary of terms is provided at the end of this section.

SECTION A - GENERAL INFORMATION

1. PROJECT ____________________
2. DATE ____________________
3. INSPECTOR ____________________
4. PRODUCT ____________________
5. SOURCE DOCUMENTATION:

____
____
____
____
____
____
____

6. INSPECTOR COMMENTS:

SECTION B - METRIC QUESTIONS

AM.1(3) d. When an error condition is detected, is resolution of the error determined by the calling unit? Y N N/A

AM.2(7) d. Is a check performed before processing begins to determine that all data is available? Y N N/A

AP.1(1) d. Is the unit free from specific references to the data base management scheme (e.g., all data calls for data base information are processed through an executive)? Y N N/A

AP.2(1) d. How many parameters in the argument list for the unit? ____ N/A

e. How many global variables are referenced by the unit? ____ N/A

f. Calculate d/(d+e) and enter score. ____ N/A

AP.2(2) d. Do the comments for global data explain where the data is derived, the data's composition, and how the data is used? Y N N/A

AP.2(3) d. Is the logical processing free from specific data storage and retrieval references (e.g., data symbolically defined and referenced)? Y N N/A

AP.2(4) d. Do the comments for all parameter input and output and local variables explain the composition and use of each data item? Y N N/A

AP.3(1) d. Is the unit free from specific references to the computer architecture? Y N N/A

AP.3(2) d. How many lines of source code, excluding comments? ____ N/A

e. How many non-HOL lines of code, excluding comments (e.g., assembly language)? ____ N/A

	f.	Calculate e/d and enter score.	[ ] N/A
AP.4(1)	d.	Is the unit free from microcode instruction statements?	Y N N/A
AT.1(1)	d.	Are all variable dimensions and sizes of dynamic arrays defined parametrically? (Note that this question must be answered N rather than N/A where the language does not provide for parametric definition.)	Y N N/A
AT.2(1)	d.	Are all accuracy, convergence, timing attributes, and timing limitations defined parametrically?	Y N N/A
AT.2(2)	d.	Are tables used in a manner which would ease the task of changing or expanding capability?	Y N N/A
CP.1(2)	d.	How many data references are identified?	[ ] N/A
	e.	How many identified data references are documented with regard to source, meaning, and format?	[ ] N/A
	f.	Calculate e/d and enter score.	[ ] N/A
CP.1(4)	d.	How many data references are identified?	[ ] N/A
	e.	How many identified data references are computed or obtained from an external source (e.g., referencing global data with preassigned values, input parameters with preassigned values)?	[ ] N/A
	f.	Calculate e/d and enter score.	[ ] N/A
CP.1(9)	d.	Are all conditions and alternative processing options defined for each decision point?	Y N N/A
CP.1(10)	d.	Are all parameters in the argument list used?	Y N N/A
CS.1(2)	d.	Does the calling sequence protocol (between units) comply with established standard?	Y N N/A

CS.1(3)	d.	Does the external I/O protocol and format comply with the established standard?	Y N N/A
CS.1(4)	d.	Does the handling of errors comply with the established standard?	Y N N/A
CS.1(5)	d.	Do all references to this unit use the same, unique name?	Y N N/A
CS.2(1)	d.	Does all data representation comply with the established standard?	Y N N/A
CS.2(2)	d.	Does the naming of all data comply with the established standard?	Y N N/A
CS.2(3)	d.	Is the definition and use of all global variables in accordance with the established standard?	Y N N/A
CS.2(6)	d.	Do all references to the same data use a single, unique name?	Y N N/A
EP.1(2)	d.	How many loops in this unit (while, repeat until, and iteration loops)?	N/A
	e.	How many loops contain non-loop dependent statements (e.g., initializing a non-loop dependent variable)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.1(3)	d.	How many units are required to be optimized for processing efficiency?	N/A
	e.	How many units are optimized for processing efficiency (i.e., compiled using an optimizing compiler or coded in assembly language)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.1(4)	d.	How many instances of two of more operations in an expression (i.e., compound expression)?	N/A

	e.	How many compound expressions are recalculated needlessly (all variables in the expression have not been reassigned values)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.1(6)	d.	How many instances of bit/byte packing/unpacking are performed?	N/A
	e.	How many instances of bit/byte packing/unpacking are performed needlessly within a loop (could be performed outside the loop)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.2(4)	d.	How many arithmetic expressions?	N/A
	e.	How many arithmetic expressions with different sized components in the same expression (e.g., bytes/word/doubleword)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.2(5)	d.	How many arithmetic expressions?	N/A
	e.	How many arithmetic expressions with mixed data types in the same expression (e.g., integer/real/boolean/literal)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
EP.2(7)	d.	How many data items (e.g., arrays, constants, variables)?	N/A
	e.	How many data items are modified?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A

ES.1(6)	d.	Are there any data packing operations?	Y N N/A
FS.1(1)	d.	Does this unit perform a single function?	Y N N/A
FS.1(2)	d.	Is a description of the function(s) provided in the comments?	Y N N/A
GE.2(2)	d.	Is this unit free from machine-dependent operations?	Y N N/A
GE.2(3)	d.	Is this unit free from strict limitations on the volume of data items it processes (e.g., data volume limits are parameterized)?	Y N N/A
GE.2(4)	d.	Is this unit free from strict limitations on the values of input data (e.g., no error tolerances are specified; no range tests or reasonableness checks are performed)?	Y N N/A
ID.1(1)	d.	How many lines of code, excluding comments?	N/A
	e.	How many references to system library routines, utilities, or other system-provided facilities?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
ID.1(3)	d.	Is the software free from any non-standard constructs of the implementation language?	Y N N/A
MO.1(3)	d.	Are the estimated lines of source code for this unit 100 lines	

		or less, excluding comments?	Y N N/A
MO.1(4)	d.	How many parameters are there in the calling sequence?	___ N/A
	e.	How manycalling sequence parameters are control variables (e.g., select an operating mode or submode, direct the sequential flow, directly influence the function of the software)?	___ N/A
	f.	Calculate e/d and enter score.	___ N/A
MO.1(5)	d.	Is all input data passed into the unit through calling sequence parameters (i.e., no data is input through global areas or input statements)?	
MO.1(6)	d.	Is output data passed back to the calling unit (e.g., through calling sequence parameters of global areas)?	Y N N/A
MO.1(7)	d.	Is control always returned to the calling unit when execution is completed?	Y N N/A
MO.1(8)	d.	Is temporary storage (i.e., workspace reserved for intermediate or partial results) used only by this unit during execution (i.e., is not stored with other units)?	Y N N/A
MO.1(9)	d.	Does this unit have a single processing objective (i.e., all processing within this unit is related to the same objective)?	Y N N/A
MO.2(5)	d.	What is the cohesion value of this unit?	___ N/A
SD.1(1)	d.	How many lines of source code, excluding comments?	___ N/A
	e.	How many non-blank lines of comments?	___ N/A
	f.	How many lines of source code with embedded comments?	___ N/A

	g.	Calculate (e+f)/d and enter score.	☐ N/A
SD.2(1)	d.	Are there prologue comments which contain all information in accordance with the established standard?	Y N N/A
SD.2(2)	d.	Are the identification and placement of comments in accordance with the established standard?	Y N N/A
SD.2(3)	d.	Are all decision points and transfers of control commented?	Y N N/A
SD.2(4)	d.	Is all machine-dependent code commented?	Y N N/A
SD.2(5)	d.	Are all non-standard HOL statements commented?	Y N N/A
SD.2(6)	d.	Are the attributes (i.e., usage, properties, units of measure, etc.) of all declared variables described by comments?	Y N N/A
SD.2(7)	d.	Do all the comments related to operations describe the purpose or intent of the operation (e.g., comment states "increment table look up-index" rather than "increment A by 1")?	Y N N/A
SD.2(8)	d.	Are the range of values and the default conditions associated with all input parameters described by comments?	Y N N/A
SD.3(1)	d.	Is the unit coded using only a higher order language?	Y N N/A
SD.3(2)	d.	Are all variable names descriptive of the physical or functional property they represent (e.g., variable names "XCOORD, YCOORD" rather then "A1, A2")?	Y N N/A
SD.3(3)	d.	Is all the code logically blocked and indented?	Y N N/A
SD.3(4)	d.	How many lines of source code, excluding comments?	☐ N/A
	e.	How many lines of source code containing more than one statement?	☐ N/A

	f.	How many continuation lines of code?	☐ N/A
	g.	Calculate 1-((e+f)/d) and enter score.	☐ N/A
SD.3(5)	d.	Is the unit structured in the standard format?	Y N N/A
SD.3(6)	d.	Are all language keywords used only with their predefined meaning (e.g., no keywords are also used as variable names)?	Y N N/A
SI.1(2)	d.	Is the unit independent of the source of the input and the destination of the output?	Y N N/A
SI.1(3)	d.	Is the unit independent of knowledge of prior processing?	Y N N/A
SI.1(4)	d.	Does the unit description/prologue include input, output, processing, and limitations?	Y N N/A
SI.1(5)	d.	How many entrances into the unit?	☐ N/A
	e.	How many exits from the unit?	☐ N/A
	f.	Calculate (1/d + 1/e)(x ½) and enter score.	☐ N/A
SI.3(1)	d.	How many conditional branch statements are there (e.g., IF, WHILE, REPEAT, DO/FOR LOOP, CASE)?	☐ N/A
	e.	How many unconditional branch statements are there (e.g., GO TO, CALL, RETURN)?	☐ N/A
	f.	Calculate 1/(1+d+e) and enter score.	☐ N/A
SI.4(1)	d.	Is the flow of control from top to bottom (i.e., flow of control does not jump erratically)?	Y N N/A
SI.4(2)	d.	How many lines of estimated source code, excluding comments?	☐ N/A

	e.	How many negative boolean and compound boolean expressions are used?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
SI.4(3)	d.	How many loops (e.g., WHILE, DO/FOR, REPEAT)?	N/A
	e.	How many loops with unnatural exits (e.g., jumps out of loop, return statement)?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
SI.4(4)	d.	How many iteration loops (i.e., DO/FOR loops)?	N/A
	e.	In how many iteration loops are indices modified to alter fundamental processing of the loop?	N/A
	f.	Calculate 1-(e/d) and enter score.	N/A
SI.4(5)	d.	Is the unit free from all self-modification of code (i.e., does not alter instructions, overlays of code, etc.)	Y N N/A
SI.4(6)	d.	How many lines of source code, excluding comments?	N/A
	e.	How many statement labels, excluding labels for format statements?	N/A
	f.	Calculate, 1-(e/d) and enter score.	N/A
SI.4(7)	d.	What is the maximum nesting level?	N/A
	e.	Calculate 1/d and enter score.	N/A
SI.4(8)	d.	How many lines of source code, excluding comments?	N/A
	e.	How many branches, conditional and unconditional?	N/A

	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
SI.4(9)	d.	How many lines of source code, excluding comments?	[ ] N/A
	e.	How many data declaration statements?	[ ] N/A
	f.	How many data manipulation statements?	[ ] N/A
	g.	Calculate 1-((e+f)/d) and enter score.	[ ] N/A
SI.4(10)	d.	How many total data items, local and global, are used?	[ ] N/A
	e.	How many data items are used locally (e.g., variables declared locally and value parameters)?	[ ] N/A
	f.	Calculate e/d and enter score.	[ ] N/A
SI.4(11)	d.	How many lines of source, excluding comments?	[ ] N/A
	e.	How many total data items, local and global are used?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
SI.4(12)	d.	Does each data item have a single use (e.g., each array serves only one purpose)?	Y N N/A
SI.4(13)	d.	Is this unit coded according to the required programming standard?	Y N N/A
SI.5(1)	d.	How many data items are used as input?	[ ] N/A
	e.	Calculate 1/(1+d) and enter score.	[ ] N/A
SI.5(2)	d.	How many data items are used for output?	[ ] N/A
	e.	How many parameters in the unit's calling sequence return output values?	[ ] N/A

	f.	Calculate e/d and enter score.	[ ] N/A
SI.5(3)	d.	Does the unit perform a single, nondivisible function?	Y N N/A
SI.6(1)	d.	How many unique operators?	[ ] N/A
	e.	How many unique operands?	[ ] N/A
	f.	How many total operands?	[ ] N/A
	g.	Calculate 1-((2 x e)/(d x f)) and enter score.	[ ] N/A
ST.1(1)	d.	How many data items are in this unit's interface (i.e., data items used to input or output data)?	[ ] N/A
	e.	Calculate 1/(1 + d) and enter score.	[ ] N/A
ST.1(2)	d.	How many global data items in this unit's interface are not adequately commented (i.e., lack comment regarding the purpose, type, or limitations)?	[ ] N/A
	e.	Calculate e/d and enter score.	[ ] N/A
ST.1(3)	d.	How many data items are in the unit's interface?	[ ] N/A
	e.	How many interface data items are in the unit with negative qualification logic (e.g., boolean values that return "TRUE" upon failure rather than success)?	[ ] N/A
	f.	Calculate 1-(e/d) and enter score.	[ ] N/A
ST.1(4)	d.	How many data items are in this unit's interface?	[ ] N/A
	e.	Calculate 1/(1 + d) and enter score.	[ ] N/A
ST.1(5)	d.	Is the unit interface established solely by arguments	

		in the calling sequence parameter list.	Y N N/A
ST.2(1)	d.	How many unique execution paths are in the unit?	N/A
	e.	Calculate 1/d and enter score.	N/A
ST.2(2)	d.	How many conditional branch statements are there (e.g., IF, WHILE, REPEAT, CASE).	N/A
	e.	Calculate 1/(1 + d) and enter score.	N/A
ST.2(3)	d.	How many other units are called by this unit (e.g., calls to other functions, subroutines, and procedures)?	N/A
	e.	Calculate 1/(1 + d) and enter score.	N/A
ST.2(4)	d.	How many iteration loops are there in the unit (e.g., DO, FOR loops)?	N/A
	e.	Calculate 1/(1 +d) and enter score.	N/A
ST.2(5)	d.	Are there comments regarding the units called by this unit and the units which call this unit?	Y N N/A
ST.3(3)	d.	Is temporary storage (i.e., work space reserved for intermediate or partial results) used only by this unit during execution (i.e., is not shared with other units)?	N/A
ST.4(1)	d.	How many global data items are used in the unit?	N/A
	e.	How many parameters are in this units calling sequence parameter list?	N/A
	f.	Calculate e/d and enter score.	N/A

ST.4(2)	d.	How many global data items are used in this unit?	[ ] N/A
	e.	Calculate 1/(1 + d) and enter score.	[ ] N/A
ST.4(4)	d.	Does this unit have a single entrance (i.e., all units calling this unit must enter at the same location)?	Y N N/A
ST.4(5)	d.	Does this unit's communication with all interfacing units pass only data parameters (i.e., does not pass any control elements)?	Y N N/A
ST.5(1)	d.	Is the unit free from unnecessarily recomputing the same value?	Y N N/A
ST.5(2)	d.	Is the unit free from statements which are never executed?	Y N N/A
ST.5(3)	d.	Is the meaning of each data item consistent throughout the unit (i.e., the use associated with each data item does not change)?	Y N N/A
ST.5(4)	d.	Is the unit free from unnecessary intermediate data items?	Y N N/A
VS.1(1)	d.	How many execution paths are there?	[ ] N/A
	e.	How many execution paths are tested?	[ ] N/A
	f.	Calculate e/d and enter score.	[ ] N/A
VS.1(2)	d.	How many total input parameters are there?	[ ] N/A
	e.	How many input parameters are tested?	[ ] N/A
	f.	Calculate e/d and enter score.	[ ] N/A

GLOSSARY

Argument List: A list of data elements that specify the input and output parameters used during execution of a software unit.

Cohesion Value: The type of relationship that exists among the elements of each software entity (Function, CSCI, Unit). The following are relative values for seven types of cohesion.

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohesion.

1) Coincidental
   - o No meaningful relationships among the elements of an entity
   - o Difficult to describe the module's function(s).
2) Logical
   - o Entity (at each invocation) one of a class of related functions (e.g., "edit all data").
   - o Entity performs more than one function.
3) Classical
   - o Entity performs one of a class of functions that are related in time (Program procedure).
   - o Entity performs more than one function.
4) Procedural
   - o Entity performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).
5) Communicational
   - o Entity has procedural strength; in addition, all of the elements "communicate" with one other (e.g., reference same data or pass data

GLOSSARY (Continued)

among themselves).

- o All functions use the same data.

6) Informational
   - o Entity performs multiple functions where the functions (entry points in the unit) deal with a single data structure.
   - o Physical packaging together of two or more entities having functional strength.
   - o All functions use the same data.

7) Functional
   - o All entity elements are related to the performance of a single function.

Control Elements: Any data items that select an operating mode or submode in the software unit, direct the sequential flow, or otherwise directly influence the function of the unit.

Control Variables: Any data items that select an operating mode or submode in the software uit, direct the sequential flow, or otherwise directly influence the function of the unit.

Coupling: The type of relationship that exists between two software entities (Functions, CSCIs, Units). In achieving a highly modular design it is essential to minimize the relationships among software entities. The goal is to design software entities with low coupling. The scale of coupling from worst to best is: 1) Content Coupling, 2) Common Coupling, 3) External Coupling, 4) Control Coupling, 5) Stamp Coupling, and 6) Data Coupling.

1) Content Coupling - One software entity references the contents of another software entity.

2) Common Coupling - Software entities reference a shared global data structure.

3) External Coupling - Software entities reference the same externally declared symbol.

GLOSSARY (Continued)

4) Control Coupling - One software entity passes control elements as arguments to another software entity.

5) Stamp Coupling - Two software entities reference the same data structure, which is not global.

6) Data Coupling - One software entity calls another and the software entities are not coupled as defined above (in 1 through 5).

Data Element: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Item: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Data Reference: A specific entity of data (e.g., variable, constant, coefficient, etc.).

Design Representation: A formal statement of the details or organization of a design using one of a number of design representation methodologies, such as, Flow Charts, HIPO Charts, PDL, etc.

Halstead's Level of Difficulty: The metric is based on Halstead's concept of the level of difficulty. A program with a high value of difficulty is likely to be more difficult to construct and this may lead to more errors. The level of difficulty is a measure of "error-proneness". Programming difficulty increases if additional operators are introduced and if an operand is used repetitively.

Lines of Code: The number of lines of source code, excluding comment lines and blank lines.

Microcode Instructions: A low-level computer instruction specifying a single machine operation.

Multiple Transfer Index Parameter: A value used to select a variation in the order of code execution (i.e., case statement, program switch, etc.).

Range-Test: A test performed to validate the object of interest over the complete spectrum of applicable values.

Subscript Value: A value used to reference an entity from a group of related objects (i.e., table index, array index, etc.).

APPENDIX B
FACTOR SCORESHEETS

Appendix B contains thirteen factor scoresheets–one for each quality factor. Each scoresheet is used to score metric elements using worksheet answers and to score the parent metrics, criteria, and factor. A factor scoresheet summary is shown on the next page that includes criteria applicable to each factor.

FACTOR SCORESHEET SUMMARY

ACQUISITION CONCERN	FACTOR SCORESHEET		APPLICABLE CRITERIA
PERFORMANCE	EFFICIENCY	(EF)	EC, EP, ES
	INTEGRITY	(IG)	SS
	RELIABILITY	(RL)	AC, AM, SI
	SURVIVABILITY	(SV)	AM, AU, DI, RE MO
	USABILITY	(US)	OP, TN
DESIGN	CORRECTNESS	(CR)	CP, CS, TC
	MAINTAINABILITY	(MA)	CS, VS, MO, SD SI
	VERIFIABILITY	(VE)	VS, MO SD, SI
ADAPTATION	EXPANDABILITY	(EX)	AT, GE VR MO, SD SI
	FLEXIBIILTY	(FX)	GE, MO, SD SI
	INTEROPERABILITY	(IP)	CL, FO, ID SY MO
	PORTABILITY	(PO)	ID, MO, SD
	REUSABILITY	(RU)	AP, DO, FS, GE ID ST MO, SD, SI

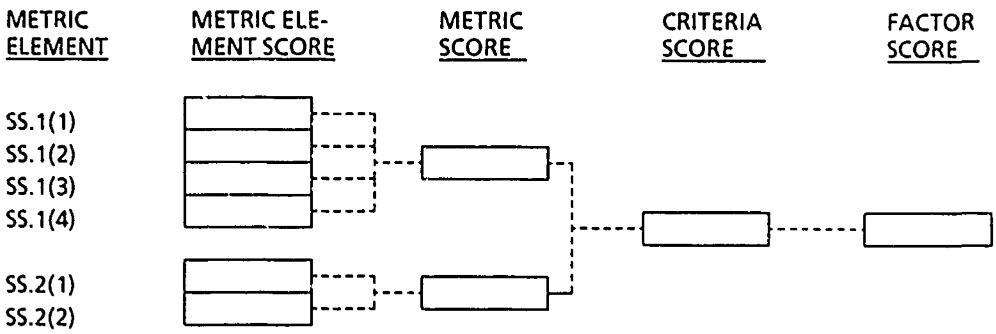
FACTOR SCORESHEET – EFFICIENCY

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
EC.1(1)				
EP.1(1)				
EP.1(2)				
EP.1(3)				
EP.1(4)				
EP.1(5)				
EP.1(6)				
EP.2(1)				
EP.2(2)				
EP.2(3)				
EP.2(4)				
EP.2(5)				
EP.2(6)				
EP.2(7)				
ES.1(1)				
ES.1(2)				
ES.1(3)				
ES.1(4)				
ES.1(5)				
ES.1(6)				
ES.1(7)				
ES.1(8)				

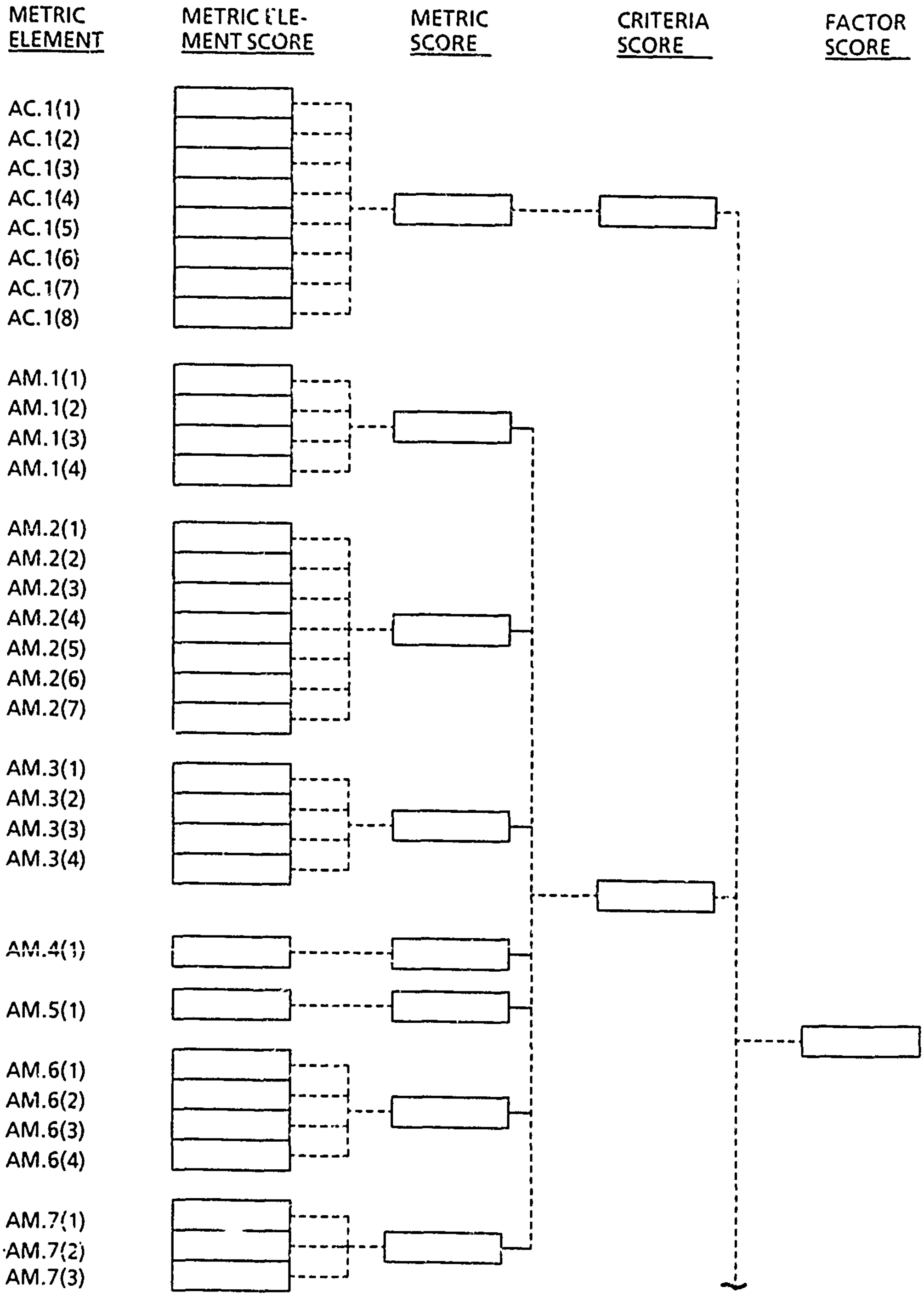
FACTOR SCORESHEET – INTEGRITY

PHASE



FACTOR SCORESHEET – RELIABILITY

PHASE



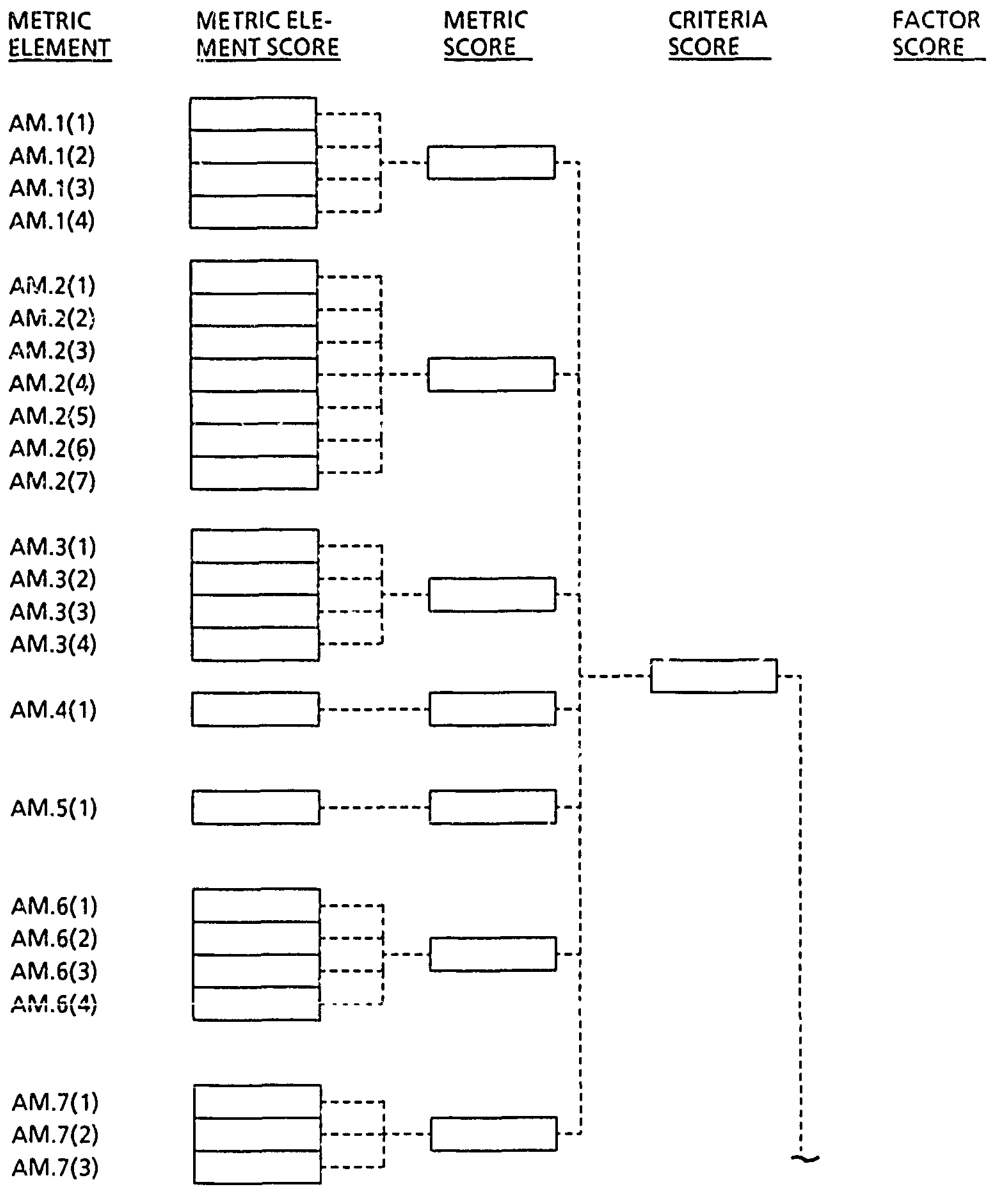
FACTOR SCORESHEET – RELIABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
SI.1(1)				
SI.1(2)				
SI.1(3)				
SI.1(4)				
SI.1(5)				
SI.1(6)				
SI.1(7)				
SI.1(8)				
SI.1(9)				
SI.1(10)				
SI.2(1)				
SI.3(1)				
SI.4(1)				
SI.4(2)				
SI.4(3)				
SI.4(4)				
SI.4(5)				
SI.4(6)				
SI.4(7)				
SI.4(8)				
SI.4(9)				
SI.4(10)				
SI.4(11)				
SI.4(12)				
SI.4(13)				
SI.4(14)				
SI.5(1)				
SI 5(2)				
SI.5(3)				
SI.6(1)				

FACTOR SCORESHEET – SURVIVABILITY

PHASE



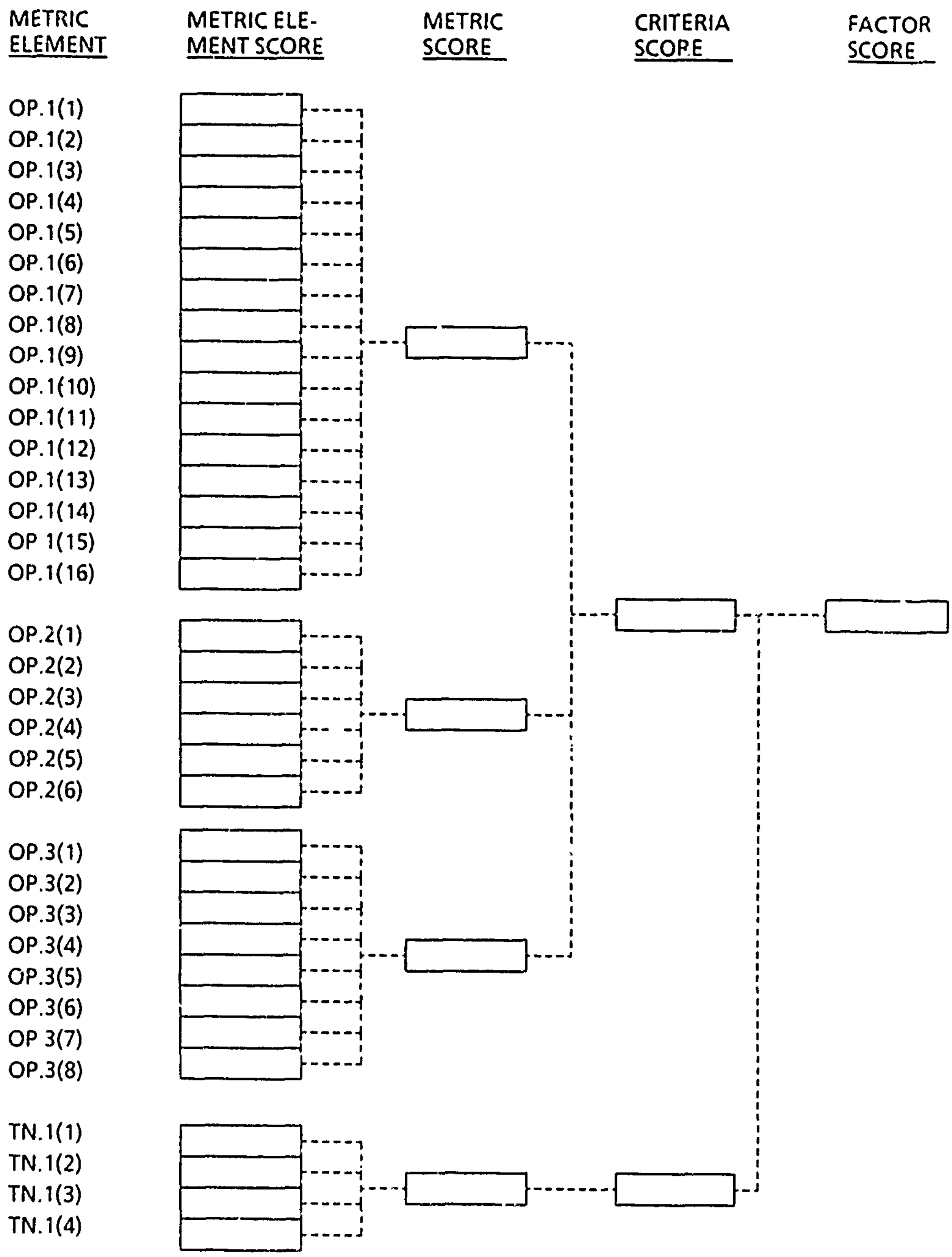
FACTOR SCORESHEET – SURVIVABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
AU.1(1)				
AU.1(2)				
AU.1(3)				
AU.1(4)				
AU.2(1)				
AU.2(2)				
DI.1(1)				
DI.1(2)				
DI.1(3)				
DI.1(4)				
DI.1(5)				
DI.1(6)				
DI.1(7)				
DI.1(8)				
DI.1(9)				
RE.1(1)				
RE.1(2)				
RE.1(3)				
RE.1(4)				
MO.1(1)				
MO.1(2)				
MO.1(3)				
MO.1(4)				
MO.1(5)				
MO.1(6)				
MO.1(7)				
MO.1(8)				
MO.1(9)				
MO.2(1)				
MO.2(2)				
MO.2(3)				
MO.2(4)				
MO.2(5)				

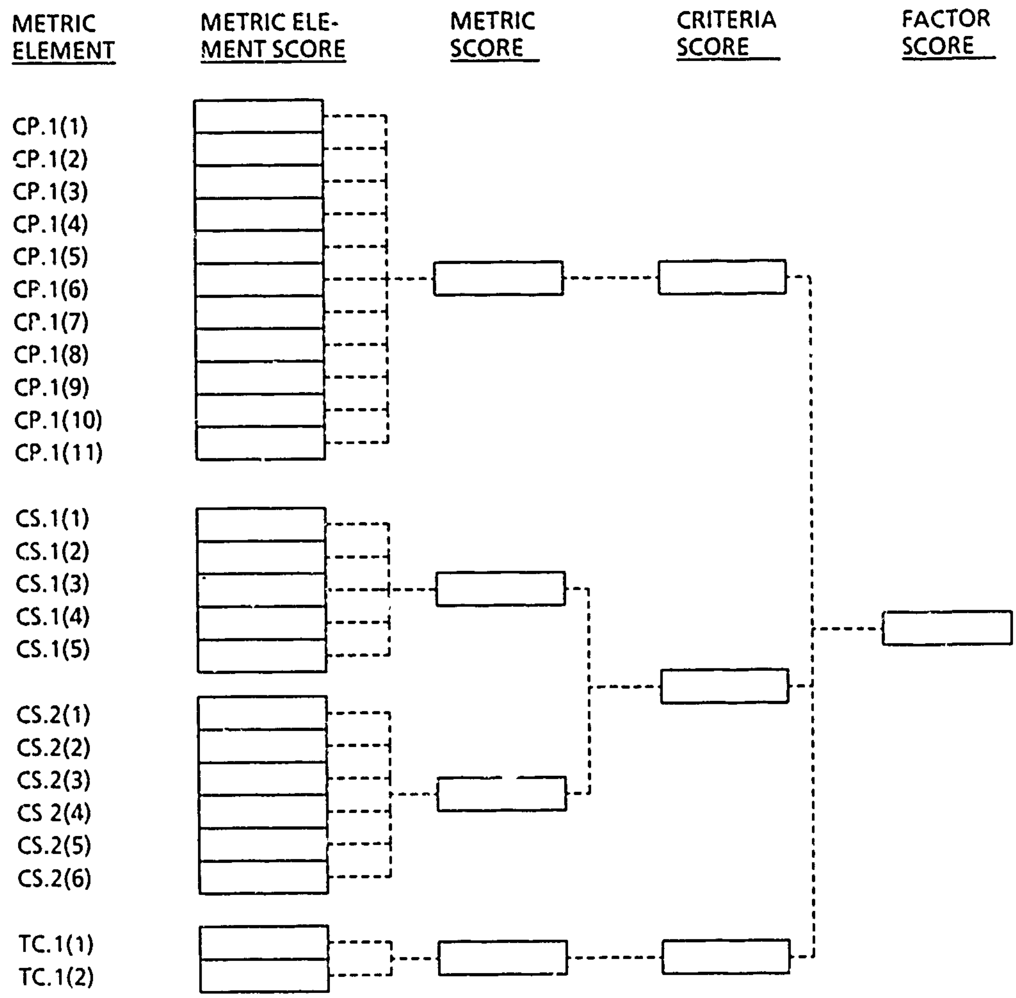
FACTOR SCORESHEET – USABILITY

PHASE



FACTOR SCORESHEET – CORRECTNESS

PHASE ☐



FACTOR SCORESHEET – MAINTAINABILITY

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
CS.1(1)				
CS.1(2)				
CS.1(3)				
CS.1(4)				
CS.1(5)				
CS.2(1)				
CS.2(2)				
CS.2(3)				
CS.2(4)				
CS.2(5)				
CS.2(6)				
VS.1(1)				
VS.1(2)				
VS.2(1)				
VS.3(1)				
VS.3(2)				
VS.3(3)				
MO.1(1)				
MO.1(2)				
MO 1(3)				
MO.1(4)				
MO.1(5)				
MO.1(6)				
MO.1(7)				
MO.1(8)				
MO.1(9)				
MO.2(1)				
MO.2(2)				
MO.2(3)				
MO.2(4)				
MO 2(5)				

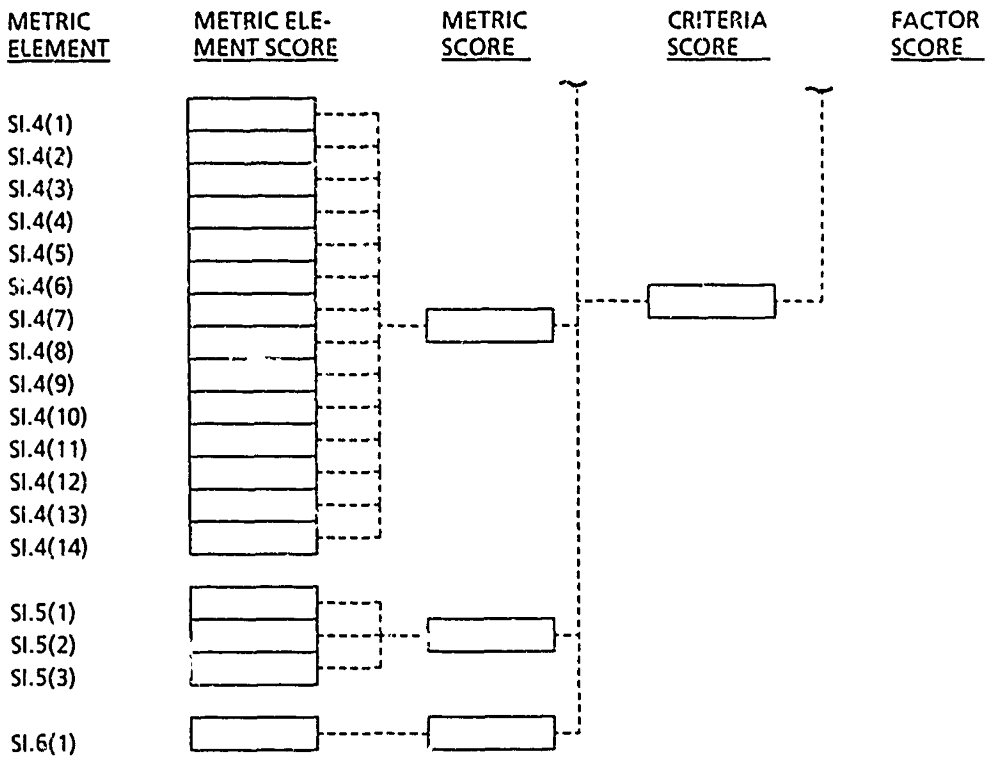
FACTOR SCORESHEET – MAINTAINABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
SD.1(1)				
SD.2(1)				
SD.2(2)				
SD.2(3)				
SD.2(4)				
SD.2(5)				
SD.2(6)				
SD.2(7)				
SD.2(8)				
SD.3(1)				
SD.3(2)				
SD.3(3)				
SD.3(4)				
SD.3(5)				
SD.3(6)				
SI.1(1)				
SI.1(2)				
SI.1(3)				
SI.1(4)				
SI.1(5)				
SI.1(6)				
SI.1(7)				
SI.1(8)				
SI.1(9)				
SI.1(10)				
SI.2(1)				
SI.3(1)				

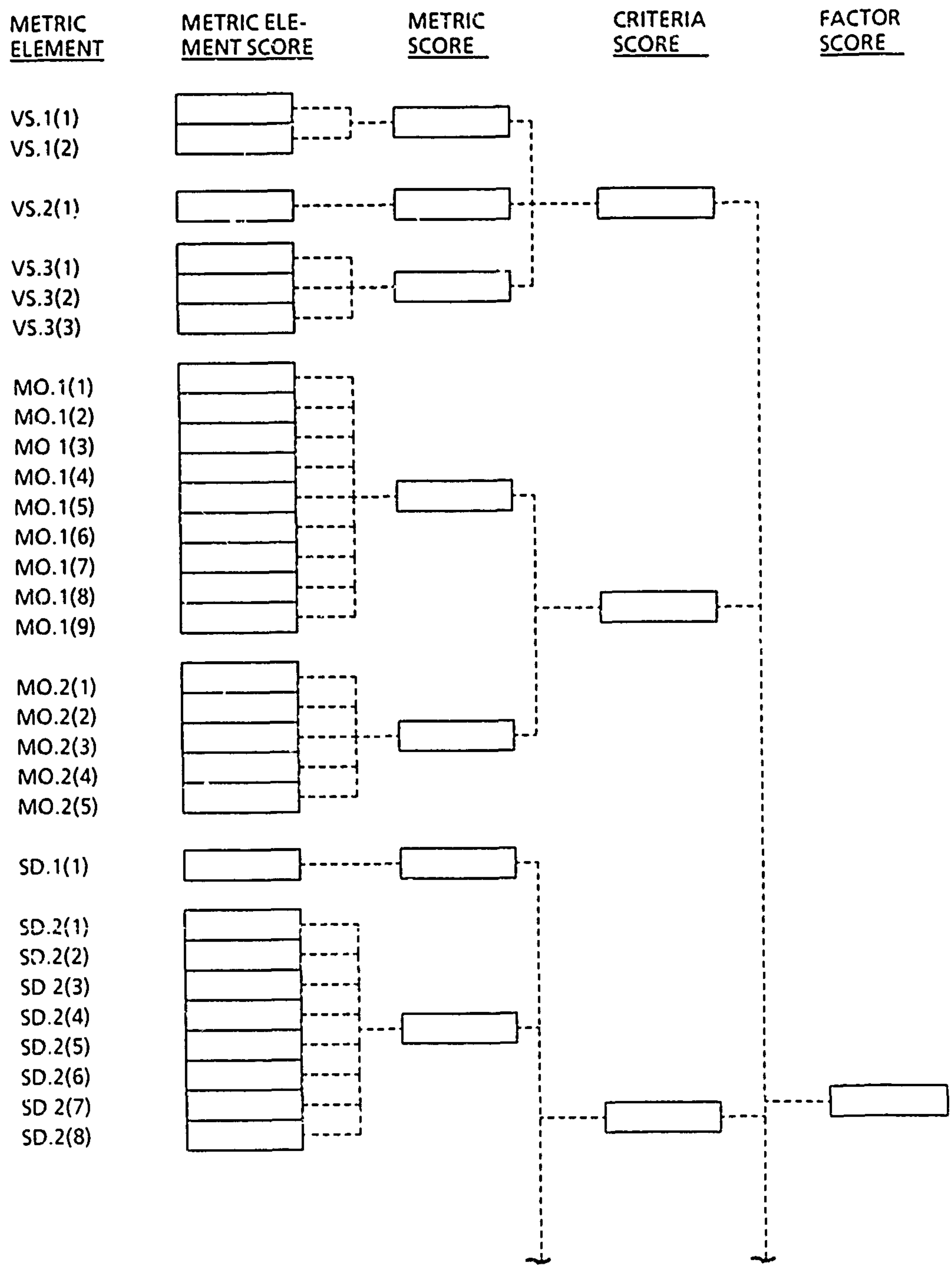
FACTOR SCORESHEET – MAINTAINABILITY (Cont.)

PHASE



FACTOR SCORESHEET – VERIFIABILITY

PHASE



FACTOR SCORESHEET – VERIFIABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
SD.3(1)				
SD.3(2)				
SD.3(3)				
SD.3(4)				
SD.3(5)				
SD.3(6)				
SI.1(1)				
SI.1(2)				
SI.1(3)				
SI.1(4)				
SI.1(5)				
SI.1(6)				
SI.1(7)				
SI.1(8)				
SI.1(9)				
SI.1(10)				
SI.2(1)				
SI.3(1)				

FACTOR SCORESHEET – VERIFIABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
SI.4(1)				
SI.4(2)				
SI.4(3)				
SI.4(4)				
SI.4(5)				
SI.4(6)				
SI.4(7)				
SI.4(8)				
SI.4(9)				
SI.4(10)				
SI.4(11)				
SI.4(12)				
SI.4(13)				
SI.4(14)				
SI.5(1)				
SI.5(2)				
SI.5(3)				
SI.6(1)				

FACTOR SCORESHEET – EXPANDABILITY

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
AT.1(1)				
AT.1(2)				
AT.1(3)				
AT.2(1)				
AT.2(2)				
AT.2(3)				
AT.3(1)				
AT.3(2)				
AT.4(1)				
AT.4(2)				
AT.4(3)				
GE.1(1)				
GE.2(1)				
GE.2(2)				
GE.2(3)				
GE 2(4)				
VR.1(1)				
MO.1(1)				
MO 1(2)				
MO 1(3)				
MO.1(4)				
MO.1(5)				
MO.1(6)				
MO.1(7)				
MO.1(8)				
MO.1(9)				
MO 2(1)				
MO.2(2)				
MO.2(3)				
MO.2(4)				
MO.2(5)				

FACTOR SCORESHEET – EXPANDABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
SD.1(1)				
SD.2(1)				
SD.2(2)				
SD.2(3)				
SD.2(4)				
SD.2(5)				
SD.2(6)				
SD.2(7)				
SD.2(8)				
SD.3(1)				
SD.3(2)				
SD.3(3)				
SD.3(4)				
SD.3(5)				
SD.3(6)				
SI.1(1)				
SI.1(2)				
SI.1(3)				
SI.1(4)				
SI.1(5)				
SI.1(6)				
SI.1(7)				
SI.1(8)				
SI.1(9)				
SI.1(10)				
SI.2(1)				
SI.3(1)				

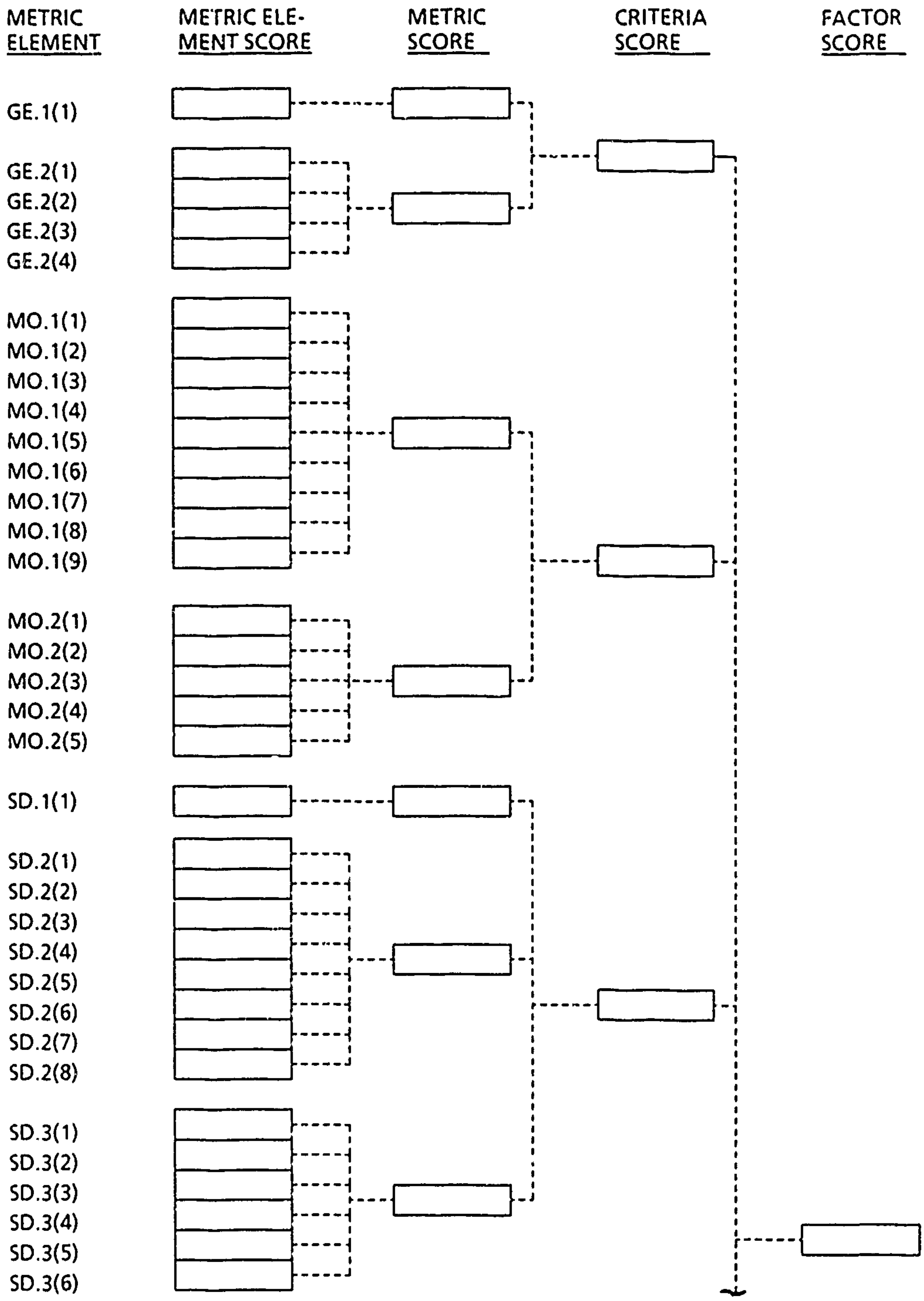
FACTOR SCORESHEET – EXPANDABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
SI.4(1)				
SI.4(2)				
SI.4(3)				
SI.4(4)				
SI.4(5)				
SI.4(6)				
SI.4(7)				
SI.4(8)				
SI.4(9)				
SI.4(10)				
SI.4(11)				
SI.4(12)				
SI.4(13)				
SI.4(14)				
SI.5(1)				
SI.5(2)				
SI.5(3)				
SI.6(1)				

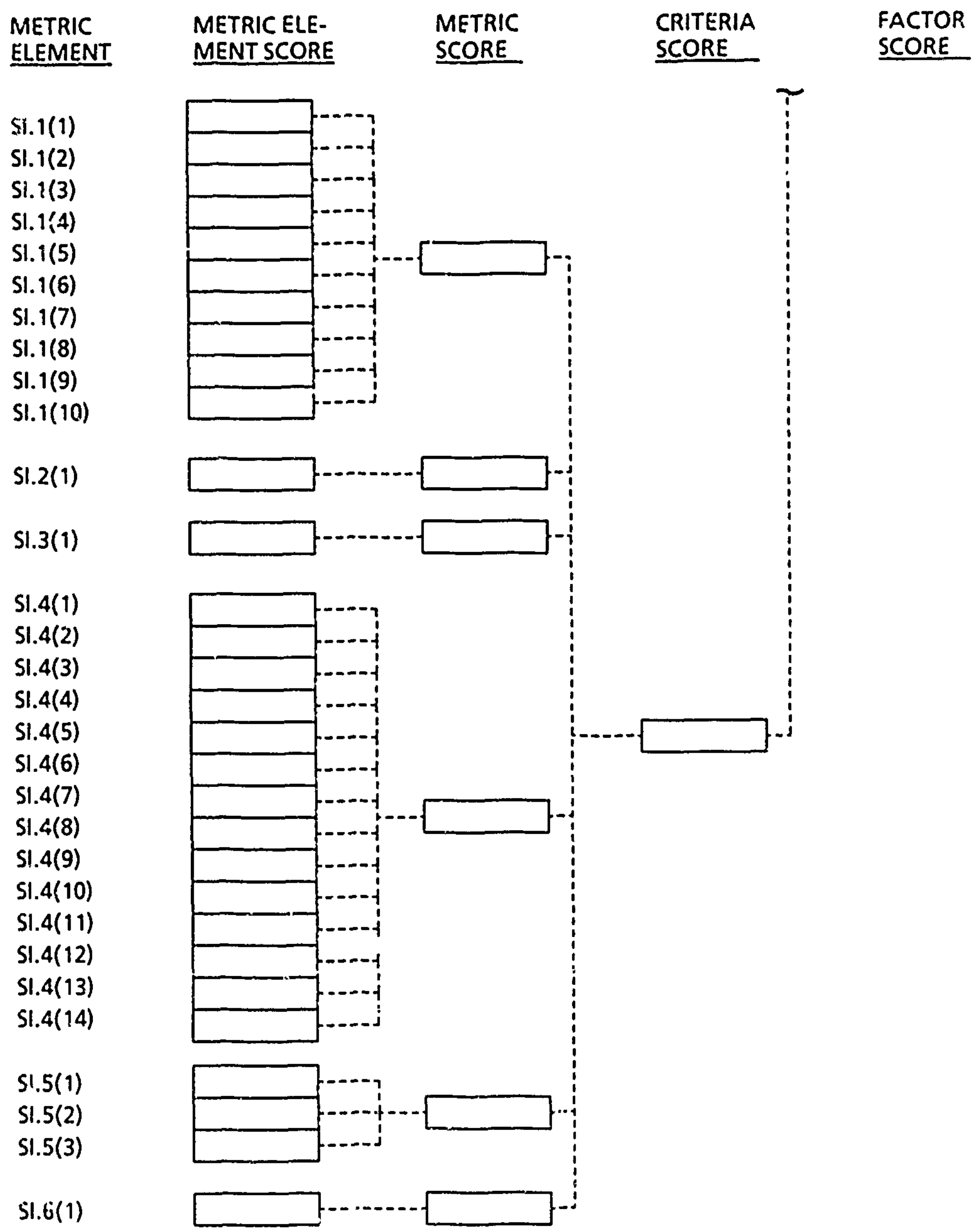
FACTOR SCORESHEET – FLEXIBILITY

PHASE



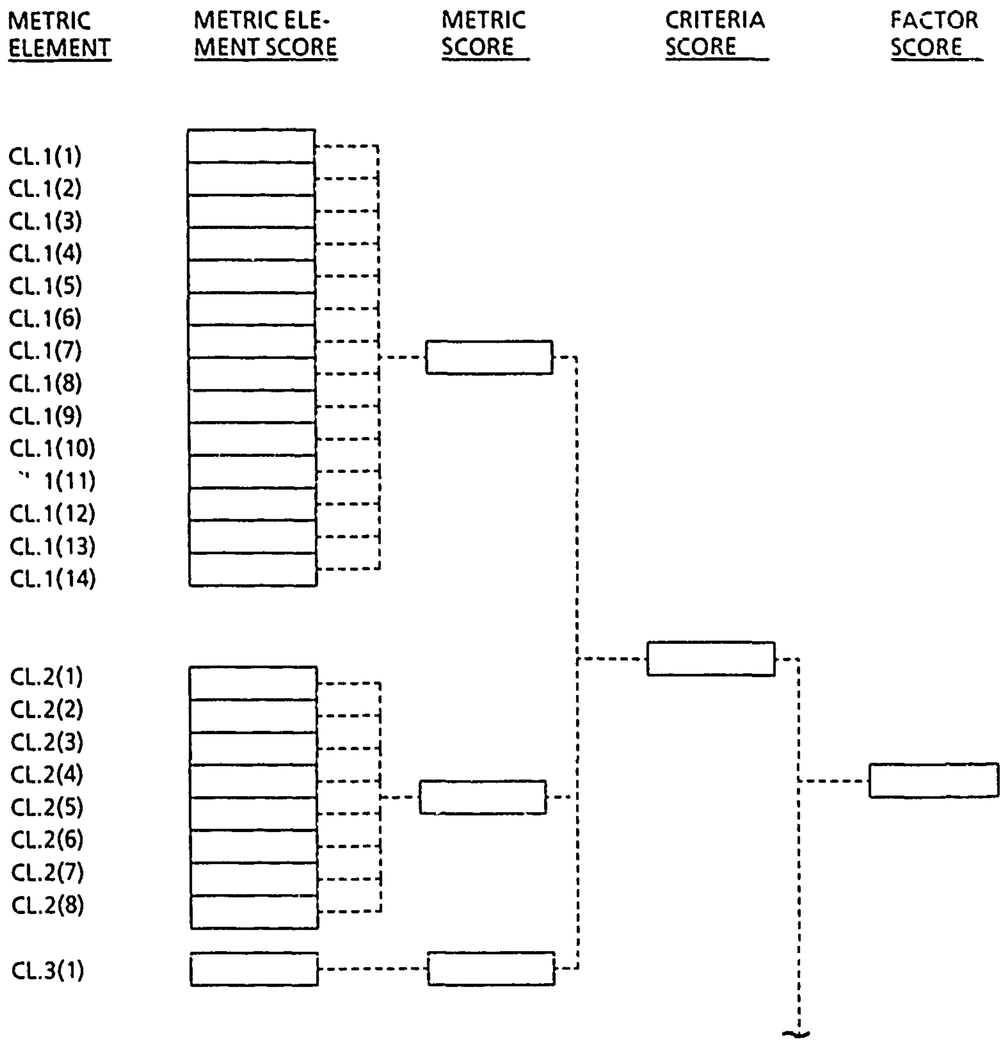
FACTOR SCORESHEET – FLEXIBILITY (Cont.)

PHASE



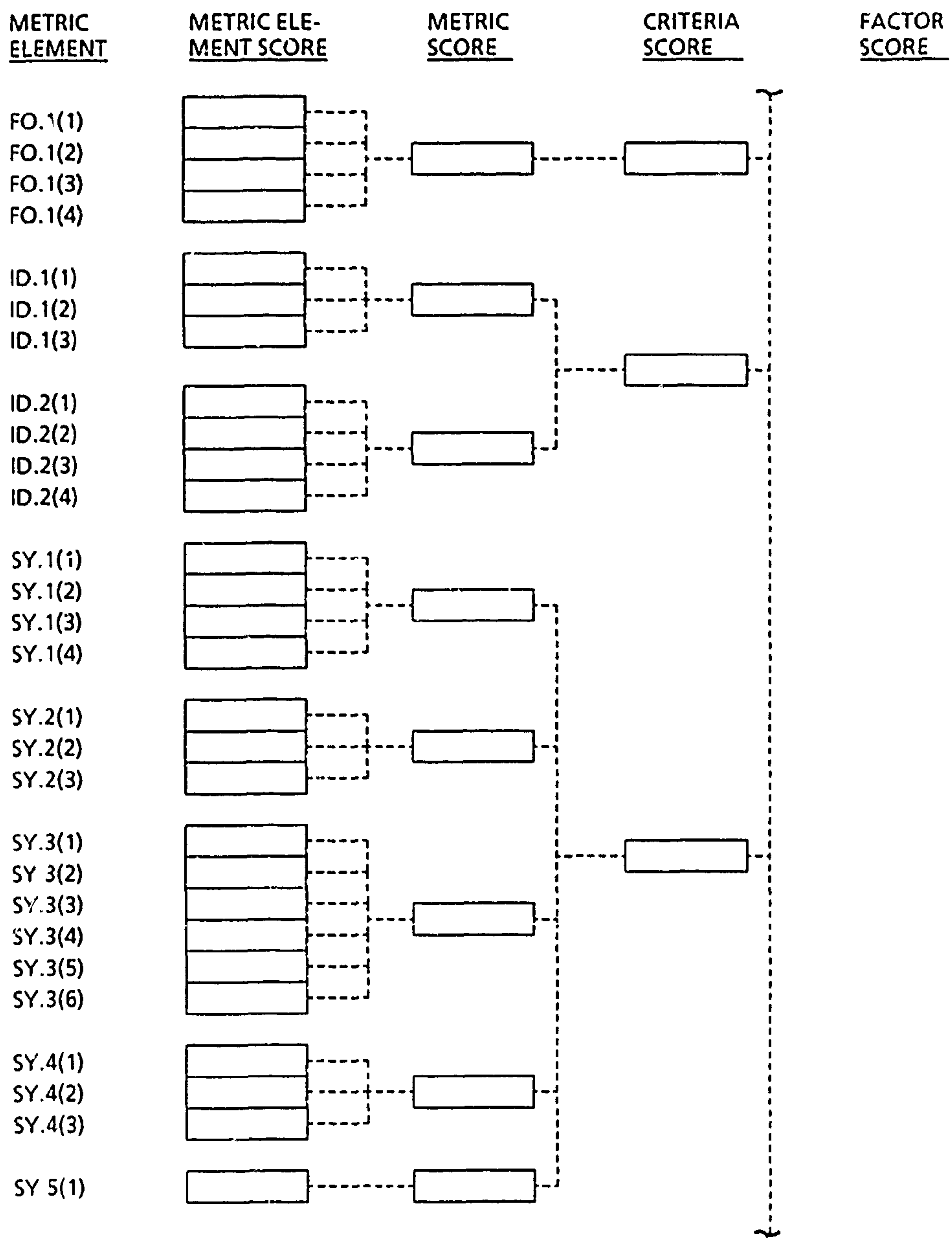
FACTOR SCORESHEET – INTEROPERABILITY

PHASE



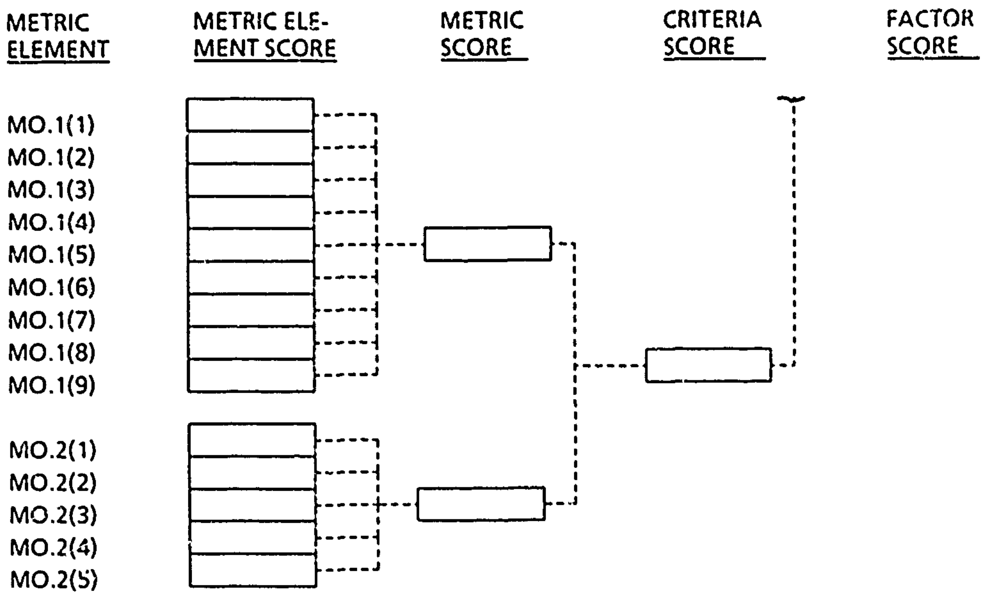
FACTOR SCORESHEET – INTEROPERABILITY (Cont.)

PHASE



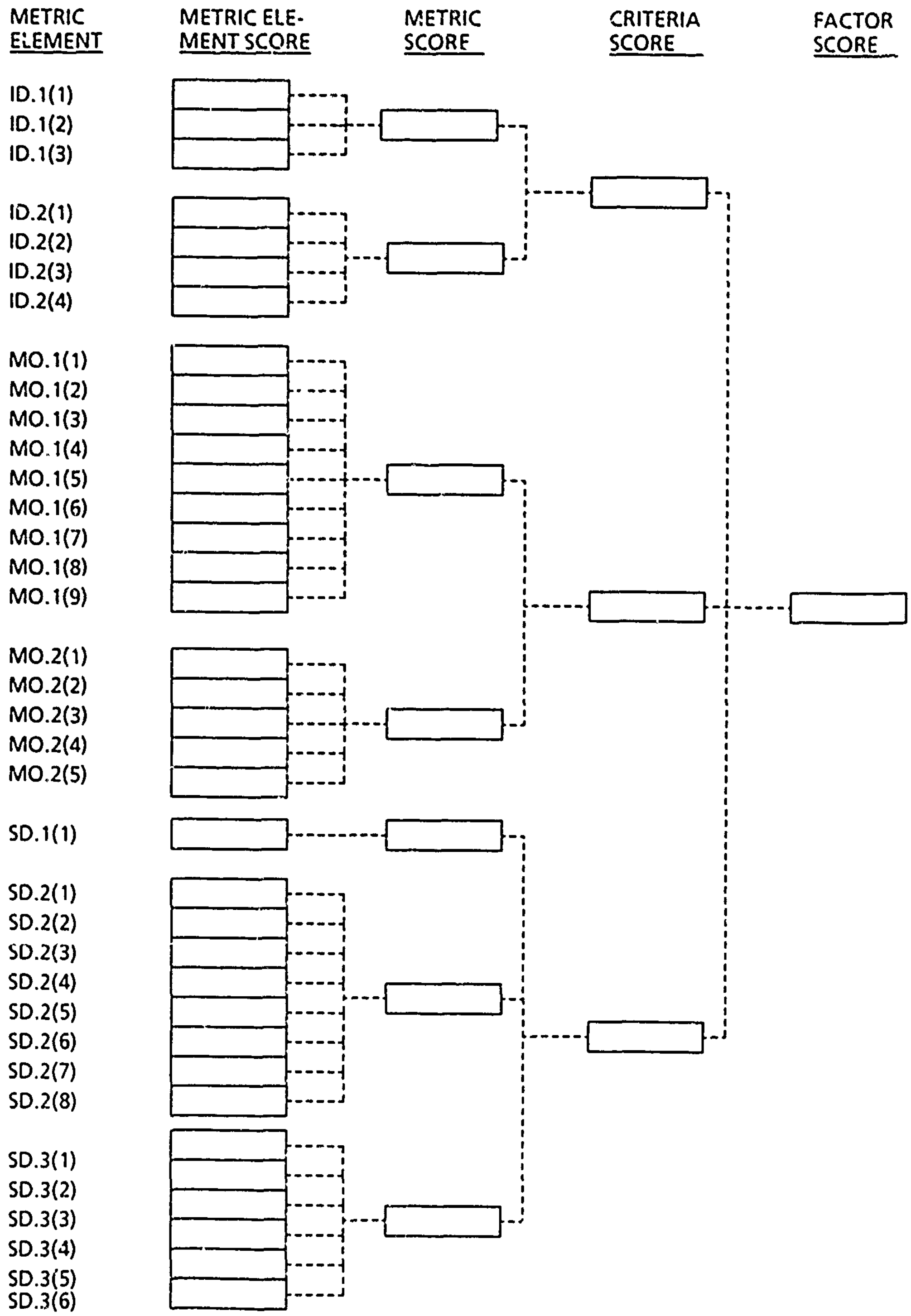
FACTOR SCORESHEET – INTEROPERABILITY (Cont.)

PHASE



FACTOR SCORESHEET – PORTABILITY

PHASE



FACTOR SCORESHEET – REUSABILITY

PHASE ☐

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
AP.1(1)				
AP.2(1)				
AP.2(2)				
AP.2(3)				
AP.2(4)				
AP.3(1)				
AP.3(2)				
AP.4(1)				
AP.5(1)				
AP.5(2)				
AP.5(3)				
DO.1(1)				
DO.2(1)				
DO.2(2)				
DO.2(3)				
DO.2(4)				
DO.2(5)				
DO.2(6)				
DO.2(7)				
DO.2(8)				
FS.1(1)				
FS.1(2)				

FACTOR SCORESHEET – REUSABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
FS.2(1)				
FS.2(2)				
FS.2(3)				
FS.2(4)				
FS.2(5)				
FS.2(6)				
FS.3(1)				
FS.3(2)				
FS.3(3)				
GE.1(1)				
GE.2(1)				
GE.2(2)				
GE.2(3)				
GE.2(4)				
ID.1(1)				
ID.1(2)				
ID.1(3)				
ID.2(1)				
ID.2(2)				
ID.2(3)				
ID.2(4)				
ST.1(1)				
ST.1(2)				
ST.1(3)				
ST.1(4)				
ST.1(5)				
ST.1(6)				
ST.2(1)				
ST.2(2)				
ST.2(3)				
ST.2(4)				
ST.2(5)				

FACTOR SCORESHEET – REUSABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
ST.3(1)				
ST.3(2)				
ST.3(3)				
ST.3(4)				
ST.3(5)				
ST.3(6)				
ST.4(1)				
ST.4(2)				
ST.4(3)				
ST.4(4)				
ST.4(5)				
ST.5(1)				
ST.5(2)				
ST.5(3)				
ST.5(4)				
MO.1(1)				
MO.1(2)				
MO.1(3)				
MO.1(4)				
MO.1(5)				
MO.1(6)				
MO.1(7)				
MO.1(8)				
MO.1(9)				
MO.2(1)				
MO.2(2)				
MO.2(3)				
MO.2(4)				
MO.2(5)				
SD.1(1)				
SD.2(1)				
SD.2(2)				
SD.2(3)				
SD.2(4)				
SD.2(5)				
SD.2(6)				
SD.2(7)				
SD 2(8)				

FACTOR SCORESHEET – REUSABILITY (Cont.)

PHASE

METRIC ELEMENT	METRIC ELEMENT SCORE	METRIC SCORE	CRITERIA SCORE	FACTOR SCORE
SD.3(1)				
SD.3(2)				
SD.3(3)				
SD.3(4)				
SD.3(5)				
SD.3(6)				
SI.1(1)				
SI.1(2)				
SI.1(3)				
SI.1(4)				
SI.1(5)				
SI.1(6)				
SI.1(7)				
SI.1(8)				
SI.1(9)				
SI.1(10)				
SI.2(1)				
SI.3(1)				
SI.4(1)				
SI.4(2)				
SI.4(3)				
SI.4(4)				
SI.4(5)				
SI.4(6)				
SI.4(7)				
SI.4(8)				
SI.4(9)				
SI.4(10)				
SI.4(11)				
SI.4(12)				
SI.4(13)				
SI.4(14)				
SI.5(1)				
SI 5(2)				
SI.5(3)				
SI.6(1)				

APPENDIX C

SOFTWARE QUALITY EVALUATION REPORT

Appendix C contains the specification of format and content for the Software Quality Evaluation Report document. Information is in data item description DID) format. The Software Quality Evaluation Report is used to describe results of metric data collection and analysis.

DATA ITEM DESCRIPTION	2. IDENTIFICATION NO(S) AGENCY	NUMBER
1. TITLE Software Quality Evaluation Report	USAF	
3. DESCRIPTION/PURPOSE The software quality evaluation report contains a quantitative assessment of achieved software quality factor levels for products released at incremental points during the software development cycle. This report is used by the Air Force to track quality levels and to assess compliance with quality factor requirements in specifications.	4. APPROVAL DATE 5. OFFICE OF PRIMARY RESPONSIBILITY 6. DDC REQUIRED 8. APPROVAL LIMITATION	
7. APPLICATION/INTERRELATIONSHIP The software quality evaluation report describes the results of metric data collection and analyses. A report is normally prepared near the end of each software development phase. Each report should contain metric data and data analyses to address each software quality factor requirement specified in the system requirements specification.	9. REFERENCES (Mandatory as cited in block 10) MCSL NUMBER(S)	

10. PREPARATION INSTRUCTIONS

1. General Requirements. The software quality evaluation report shall describe results of metric data collection and analyses. Data analyses information shall include correlation of metric scores to factor scores for each software quality factor requirement. Raw metric scores and factor scoring trends shall be included.

2. Detailed Requirements. For convenience in describing the minimum essential content, the following paragraphs show a normal format for presentation of material. In the following description, paragraph headings and numbers indicate the general nature of the topic and are minimum mandatory requirements.

a. Section 1.0 - Introduction. This section shall describe the purpose and scope of the report.

b. Section 2.0 - References. This section shall list both government and non-government references and shall include identification of system/software products used as source material for metric data collection.

c. Section 3.0 - Software Quality Evaluation Data. This section shall describe detailed results of metric data collection and analyses and shall identify variations from software quality requirements.

(1) Paragraph 3.1 - Product Source Material. This paragraph shall describe the software development phase and system/software products used as source material for collecting metric data.

DD FORM 1 JUN 68 1664 S/N-0102-019-4000 PLATE NO. 19448 PAGE 1 OF 3 PAGES

U.S. GOVERNMENT PRINTING OFFICE: 1971—714-277/1030 3-1 D-8817

(2) Paragraph 3.2 - Requirement Allocation Relationships. This paragraph shall identify and describe the derivation of relationships used for scoring based on the allocation of quality factor requirements to software elements (CSCIs and units). Formulas and lists should be used. For example, Qsf1 = (Qf1 + Qf2 + ... + Qfn)/N, where:

Qsf1 is the quality factor score for system-level function 1,
Qf1 is the quality factor score for software element 1,
Qf2 is the quality factor score for software element 2,
...
and Qfn is the quality factor score for software element n.

One formula is required for each software quality factor of each system-level function for which software quality factor requirements have been specified. This paragraph shall also identify the specific relationships (criteria and metrics to factors) which were used to calculate software quality factor scores.

(3) Paragraph 3.3 - Data Collection. This section shall describe results of metric data collection and reduction and shall include descriptions of:

(a) Selection and use of metric worksheets to collect metric element data.
(b) Selection and use of metric scoresheets to compute metric scores, criterion scores and factor scores.

(4) Paragraph 3.4 - Data Analyses. This section shall describe results of metrics data analyses and shall include descriptions of:

(a) Computation of quality factor scores for each system-level function.
(b) Comparison of metric scoring with specified quality factor requirements (goals) and analyses of variations from requirements. Causes and remedies shall be expl~~ed for each variation.
(c) Trend analyses, showing software quality factor scoring trends with respect to software development phases.

(5) Paragraph 3.5 - Recommendations. This paragraph shall provide the following:

(a) Explanations and rationale for scoring variations.
(b) Recommendations for correcting scoring variations.

d. Appendix A - Summary Information. This section shall be included as an appendix to the software quality evaluation report. It shall include textual and pictorial material to elaborate and refine material presented in section 3, Software Quality Evaluation Data. These items shall include tabular representations of:

(1) Software quality factor requirements allocation to software elements.
(2) A comparison of software quality factor scoring with specified requirements.
(3) Quality Criteria scoring for each factor.
(4) Quality Metric scoring for each criteria.

e. Appendix B - Factor Scoresheets. This section shall be included as an appendix to the software quality evaluation report. It shall contain the scoresheets with scores for all applicable factors, criteria, metrics, and metric elements.

f. Appendix C - Metric Worksheets. This section shall be included as an appendix to the software quality evaluation report. It shall contain the metric worksheets with answers to all applicable metric element questions.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, solid state sciences, electromagnetics and electronic reliability, maintainability and compatibility.